

PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.ORG

EDIÇÃO #39- FEVEREIRO 2013

ISSN 1647-0710



A PROGRAMAR

THREADS SEMÁFOROS E DEADLOCKS
O JANTAR DOS FILÓSOFOS

OPENCL PROGRAMAÇÃO

A WEB EM TEMPO REAL
NA PLATAFORMA ASP.NET

BUBBLE SORT A TÉCNICA
DA BOLHA

SEO (SEARCH ENGINE OPTIMIZATION)
PARTE III

ANÁLISES

VISUAL BASIC 2012 CURSO
COMPLETO

Mail

Pinball FX2

NO CODE

COMO COMEÇAR? **PROGRAMAR**

NOVIDADES **SHAREPOINT 2013**

COLONAS

GEOCOORDINATEWATCHER
COMO UM SERVIÇO REATIVO **C#**

COMUNIDADES

CERTIFICAÇÃO MICROSOFT

EQUIPA PROGRAMAR

Coordenador
António Santos

Editor
António Santos

Design
Sérgio Alves
Twitter: [@scorpion_blood](https://twitter.com/scorpion_blood)

Redacção
António Pedro Cunha
André Vala
Miguel Lobato
Nuno Santos
Patrício Domingues
Paulo Morgado
Ricardo Rodrigues
Rita Peres
Sara Silva
Sérgio Ribeiro
Sérgio Laranjeira
Telmo Vaz

Staff
Ana Barbosa
António Pedro Cunha
António Santos
António Silva
Fábio Domingos
Jorge Paulino
Sara Santos

Contacto
revistaprogramar@portugal-a-programar.org

Website
<http://www.revista-programar.info>

ISSN
1 647-071 0

<HTTP Error 410 Gone>

Muitas vezes deparamo-nos com mensagens de erro, das quais nem sempre entendemos logo o significado. Por vezes somos presenteados com verdadeiras “perolas da internet”, no lugar das mensagens de erro, como cheguei a ver num site por todos nos conhecido “HTTP Error 501 A bunch of highly skilled apes was sent to fix this problem”.

Quantas vezes desatamos a rir com estas mensagens, que são a verdadeira prova de que os I.T.'s têm um grande sentido de humor, mesmo nas situações mais complexas.

Um ano depois de ter assumido a direcção editorial da revista, e vendo em perspectiva todo o processo como se de um bloco “*try { (...); } Catch e as Exception { (...); }*”, se tratasse, vejo que a tecnologia evoluiu imenso, novas tecnologias apareceram, coisas espantosas aconteceram, alguns sistemas operativos saltaram “fronteiras”, as linguagens de programação evoluíram, novas funcionalidades e métodos apareceram, o “interminável ciclo” não parou e com ele, todos aprendemos mais, numa constante evolução, na tentativa de fazer cada vez melhor.

Como não deve existir *try* sem um *catch*, nós também tivemos algumas *exceptions*, que nos ajudaram a melhorar esta publicação que vos trazemos, que é feita por nós, (todos aqueles que na revista colaboram, de forma mais ou menos directa), para vós, (todos quanto lêem a revista).

Quando escolhi o título, pensei: “mais um título de mensagem de erro, qualquer dia isto pega a moda e esgota as mensagens de erro de toda a pilha de protocolos”. Depois pensei melhor e achei que o título se adequa ao momento. Neste momento todo o mundo vive uma “transição” algumas tecnologias partem, outras aparecem, as mensagens de erro não serão muito diferentes, apenas mais criativas, e o grande ciclo irá continuar.

Não tenho por habito fazer promessas, quando não tenho a certeza absoluta que as posso cumprir, por esse motivo, no final deste editorial resta-me prometer aquilo que posso cumprir: “Sempre que haja uma *exception*, estaremos cá para fazer o seu *catch*, aprendendo, adaptando, melhorando, evoluindo, fazendo aquilo que todos sabemos fazer de melhor: Criar novas soluções, resolver novos problemas, inovar, em cada passo, e dar de nós um pouco mais.

Esta é a edição em que uma iteração se completa, desta iteração levo comigo o sentido de dever cumprido. Cumprido com agrado, e com prazer, pois nunca me esqueço que antes de ser editor, fui leitor muitos anos! Espero ter cumprido as expectativas, e na nova iteração, fazer melhor, pois a tecnologia são constantes iterações, em que a seguinte é sempre melhor que a anterior.

António Santos
<antonio.santos@revista-programar.info>

A revista PROGRAMAR é um projecto voluntário sem fins lucrativos. Todos os artigos são da responsabilidade dos autores, não podendo a revista ou a comunidade ser responsável por alguma imprecisão ou erro.

Para qualquer dúvida ou esclarecimento poderá sempre contactar-nos.

TEMA DE CAPA

- [7](#) Windows 8 Store Apps – Do sonho à realidade **(Sara Silva)**

A PROGRAMAR

- [13](#) Threads, Semáforos e Deadlocks – O Jantar dos Filósofos **(Rita Peres e Diogo Barros)**
- [21](#) Programação em OpenCL **(Patrício Domingues)**
- [32](#) Accionamento de Servos Usando Arduino **(Nuno Santos)**
- [35](#) A web em tempo real na plataforma ASP.NET **(Ricardo Rodrigues)**
- [38](#) Bubble Sort – A Técnica da Bolha **(Telmo Vaz)**

COLUNAS

- [46](#) C# - O GeoCoordinateWatcher Como Um Serviço Reativo **(Paulo Morgado)**

ANÁLISES

- [50](#) Visual Basic 2012: Curso Completo **(Sérgio Ribeiro)**
- [51](#) Multimédia e Tecnologias Interativas (5ª Edição Atualizada e Aumentada) **(Sérgio Laranjeira)**

COMUNIDADES

- [53](#) NetPonto - Certificação Microsoft **(Sara Silva)**

NO CODE

- [57](#) Programar—como começar? **(António Pedro Cunha)**
- [61](#) Novidades na Pesquisa no SharePoint 2013 **(André Vala)**
- [72](#) SEO – Acessibilidades, Usabilidade e Demografia **(Autor)**
- [75](#) Entrevista a Miguel Gonçalves
- [77](#) Projecto em Destaque na Comunidade P@P: PROGRAMAR

EVENTOS

23 Fev. 2013	36ª Reunião Presencial NetPonto em Lisboa
25 Fev. a 1 de Mar. 2013	XX SINFO
26 Fev. 2013	XXXI Encontro da Comunidade SQLPort
1 e 2 de Março 2013	Google DevFest 2013
16 Março 2013	SQLSaturday

Para mais informações/eventos: http://bit.ly/PAP_Eventos. Divulga os teus eventos para o email eventos@portugal-a-programar.pt

EUA Promovem Civic Hacking Day para resolver problemas sociais.

O governo dos EUA está a apelar ao lado mais cívico dos programadores para que coloquem os seus conhecimentos técnicos e perícias informáticas ao dispor da restante sociedade. Num evento de dois dias batizado como Civic Hacking Day o objetivo é criar ferramentas online que sejam capazes de ajudar as diferentes comunidades norte-americanas.



A iniciativa vai contar com a participação da NASA, do Departamento do Trabalho, da Random Hacks of Kindness e da entidade responsável pelos censos nos EUA. Estas organizações e instituições públicas vão fornecer dados públicos compilados e desafios específicos para que determinados problemas comunitários ganhem uma resposta em pouco tempo e com um nível de maturidade aceitável.

O evento já está marcado em 27 cidades espalhadas um pouco por todo o país e vai funcionar ao estilo hackathon, onde no espaço de tempo definido tenta-se fazer o maior número possível de programas e soluções informáticas com um determinado propósito.



Boston, Chicago, Denver, Detroit, Palo Alto, São Francisco e Nova Iorque são algumas das localizações já confirmadas. "Enquanto as comunidades de hacking cívico já trabalham há muito tempo para melhorar o nosso país e o mundo, neste verão, pela primeira vez, os programadores locais de todos os cantos da nação vão reunir e partilhar a missão de resolver desafios relevantes para os nossos bairros, para as nossas vizinhanças, para as nossas cidades, para os nossos estados e para o nosso país", pode ler-se no comunicado da Casa Branca.

Na página oficial da iniciativa, hackforchange.org, é possível ver uma lista de benefícios, todos eles sociais e comunitários, que vão resultar do evento.

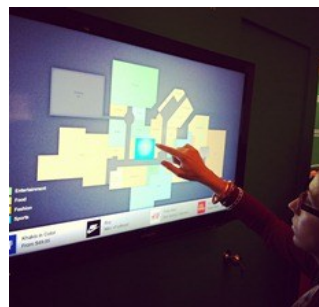
Escrito ao abrigo do novo Acordo Ortográfico

Fonte: Tek Sapo

Produto da empresa portuguesa CoVii presente na CES 2013

A start-up tecnológica portuguesa CoVii apresentou no passado dia 8 de Dezembro de 2012 na CES o seu novo software para sensores 3D que transforma qualquer ecrã de qualquer dimensão num touchscreen - ViiHaptic.

A empresa israelita PrimeSense™, que criou a tecnologia de sensores 3D, acredita que este novo software da CoVii vai revolucionar o mercado do digital signage, pois pode transformar qualquer ecrã de qualquer tamanho já usado pelas empresas, em touchscreen de uma forma robusta e com um custo muito inferior a um verdadeiro touchscreen ou tela de toque.



Este novo software da CoVii permite reconhecer o toque no ecrã, mas acrescenta-lhe ainda outras funcionalidade como o hovering, que permite controlar ações à distancia sem a necessidade do toque, e o reconhecimento do utilizador.

Desta forma o software deteta quantos utilizadores estão a interagir, em que posição se encontram, sendo uma mais-valia para a comunicação das empresas.

O ViiHaptic corre nos principais sistemas operativos (Windows, Mac e Linux), é de fácil calibração e permite ainda a inserção de conteúdos em diferentes linguagens de programação.

Este produto contém inúmeras utilizações, sendo que o facto de não existir necessidade de toque no ecrã, o torna ideal para locais onde a higiene e segurança tenham de ser garantidas (ex. blocos operatórios, controlo de maquinaria) ou até para locais em que a interação de múltiplos indivíduos com um ecrã seja frequente (ex. mobiliário urbano para informação digital em aeroportos, shoppings).

O software apresentado pela CoVii na CES permite uma inovadora forma de interação e leva o utilizador a um outro nível de interatividade.

Mais uma vez é de louvar o esforço de todos os que escolhem apostar e ficar em Portugal. Afinal, Portugal ainda vale a pena.

Escrito ao abrigo do novo Acordo Ortográfico

Fonte: CES 2013

Ubuntu chega aos smartphones já este ano

De acordo com a Canonical, o Ubuntu OS para smartphones está a ser desenvolvido dando prioridade à interface e às ferramentas que facilitam a criação de software para o novo sistema operativo. Aliás, o SDK vai permitir a criação em simultâneo de aplicações para a versão desktop e para a versão móvel do Ubuntu.

Apesar de ainda não terem sido anunciados terminais com o novo sistema operativo, a Canonical já fez saber que o Ubuntu OS para telemóveis vai ser demonstrado na CES, uma das maiores feiras de tecnologia, que vai começar já na próxima semana (8 de janeiro).

De acordo com a Cnet, o Ubuntu vai usar os mesmos drivers do Android, o que vai facilitar a adaptação do hardware.

Pode encontrar as informações sobre o novo sistema operativo em www.ubuntu.com/devices/phone.



Escrito ao abrigo do novo Acordo Ortográfico

Fonte: [exameinformática](http://exameinformatica.com)

Erro de Digitação fez com que o Surface estivesse de graça

O fato é que um pequeno erro de digitação, divulgado por um [Tweet de Tom Warren](#), fez com que o Surface estivesse disponível na página por “\$000” – isso mesmo, zero dólares. A gafe, porém, já foi corrigida: no momento em que essa notícia e escrita, o valor mostrado passou a ser de \$ 899 (R\$ 1834).

Agora, resta saber se nos EUA, o famoso “país dos processos”, isso não vai acarretar em uma enxurrada de reclamações de consumidores insatisfeitos por propaganda enganosa.

Escrito ao abrigo do novo Acordo Ortográfico
Fonte: [Tecmundo](http://tecmundo.com)

Co-fundador do Reddit, Aaron Swartz, suicida-se



Aaron Swartz, activista online e fundador do Infogami, um serviço que mais tarde se fundiu com a Reddit, cometeu suicídio na Cidade de Nova York, a 11 de Janeiro, de acordo com o Tech.

A notícia foi revelada ao Tech pelo tio de Swartz, Michael Wolf e confirmada pelo advogado de Swartz, Elliot R. Peters. “ A informação trágica e desoladora que receberam é, lamentavelmente verdade,” disse Peters.

Nascido em 1986, Swartz foi co-autor da primeira especificação de RSS aos 14 anos. Também fundou a Infogami, um serviço financiado pela Y Combinator que mais tarde se fundiu com o site de redes sociais Reddit.

Swartz também co-fundou a Demand Progress, um grupo de advocacia que realiza comícios incentivando pessoas a “tomar ação acerca das notícias que as afetam—contatando o congresso e outros líderes, financiando táticas de pressão e espalhando a palavra nas suas comunidades.”

Em Julho de 2011, Swartz foi preso por alegadamente recolher 4 milhões de trabalhos académicos do arquivo do jornal online JSTOR. Foi a tribunal em Setembro de 2012, afirmando ser inocente.

Uma postagem de blog de 2001 no website de Swartz revela uma causa possível para tirar a própria vida: depressão. Na postagem, Swartz descreve as suas experiências com depressão grave, bem como outros problemas de saúde, incluindo enxaquecas.

Escrito ao abrigo do novo Acordo Ortográfico

Fonte: mashable.com

Tradução: Sara Santos

TEMA DE CAPA

Windows 8 Store Apps – Do sonho à realidade

Windows 8 Store Apps – Do sonho à realidade

Este artigo tem como objectivo expor, de uma forma geral, o processo de desenvolvimento de aplicações para a Windows 8 Store.



No passado mês de Setembro foi anunciado, pela Microsoft, que a Windows 8 Store passava a estar “aberta” a todos os programadores, ou seja, qualquer indivíduo ou empresa passa a ter acesso à [Windows Store Dashboard](#), através de uma subscrição anual (37€ e 75€ respetivamente), a qual permite submeter, no mercado, as suas aplicação.

De salientar que todos os estudantes, através do programa [DreamSpark](#), têm acesso gratuito ao [Windows Store Dashboard](#), assim como os [subscritores MSDN](#).

A Windows Store é mais uma oportunidade de negócio que pode ser explorada! E os mais sonhadores, decerto que já imaginaram a sua aplicação na Store.

Pois bem, então e que tal tornar esse sonho realidade?

A Microsoft disponibiliza um conjunto de recursos que de uma forma geral se dividem em duas referências principais:

- [design.windows.com](#)
- [dev.windows.com](#)

Assim, antes de iniciarmos o desenvolvimento de uma aplicação devemos começar por desenhá-la, na realidade a prototipagem é cada vez mais comum nos nossos dias. Sugeria, então, que se começasse por expor numa folha de papel miniaturas de cada página e com setas se definisse o fluxo. À parte poder-se-á desenhar cada ecrã com o conteúdo principal de cada página sem entrar em grande detalhe.

Neste primeiro rascunho já será possível perceber e detectar um conjunto de questões relacionadas com a interface e o utilizador:

- Qual a melhor forma de posicionar o conteúdo?
- Qual a melhor forma de apresentar as opções ao utili-

zador?

- Qual a melhor abordagem para que o utilizador consiga ter uma boa percepção da aplicação aquando da primeira utilização?

Outras mais:

- E o “Login”/“Logout” da aplicação, onde vai ser apresentado?
- E as definições da aplicação?

Das duas referências acima mencionadas, o [design.windows.com](#) é a referência onde podemos encontrar informação relacionada com o desenho da interface e todos os conceitos a ela associados.

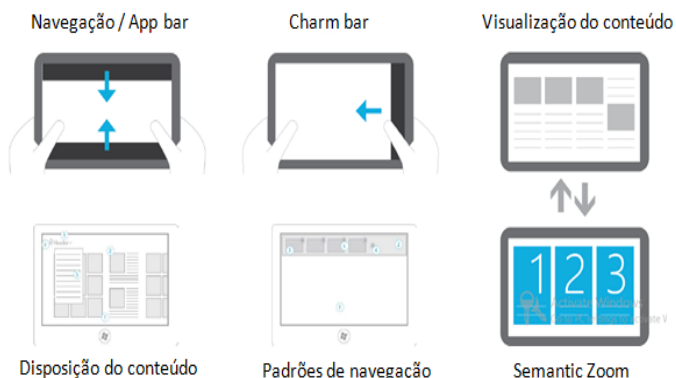
Na secção de Planeamento, podemos recolher informação que nos faz pensar sobre:

- ⇒ Foco: Qual o foco da aplicação? O que faz a minha aplicação ser genial? Com toda a certeza vão existir milhares de aplicações, mas cada aplicação tem características especiais, mas o que faz com que a nossa aplicação seja a melhor?;
- ⇒ Fluxo: Planear o fluxo da aplicação é outro aspecto a ter em conta, quanto mais complexa for a navegação mais difícil se torna a sua utilização por parte do utilizador;
- ⇒ Funcionalidades: ao determinar as funcionalidades de base, devem ser incluídos os novos conceitos do Windows 8, como por exemplo, a Partilha e a Pesquisa, de forma a tornar a aplicação mais rica e que a mesma possa transmitir uma grande integração com o próximo SO.
- ⇒ Saliento, ainda, aspectos relacionados com as várias vista (“FullScreen”, “Filled”, “Span”, “Portrait”), sendo que localização, animação das “Tiles”, notificações, “roaming” são outros conceitos abordados.
- ⇒ Ganhar dinheiro: de que forma se pode tirar partido monetário da nossa aplicação?
- ⇒ Neste tópico recomendo a leitura do artigo [EN][Selling apps](#).

TEMA DA CAPA

WINDOWS 8 STORE APPS – DO SONHO À REALIDADE

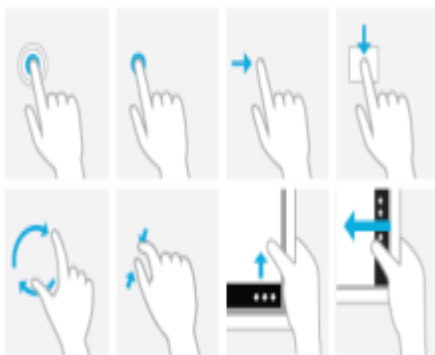
⇒ Interface, a primeira impressão e validações vão nos permitir dar uma orientação sobre o que está a ser desenhado, ou seja, perceber se a nossa aplicação vai ter boa receção por parte do utilizador e se vai ser validada ao nível do processo de certificação.



Para ajudar e complementar o planeamento da aplicação é constantemente apresentado um conjunto de guias (“UX Guidelines”), as quais nos irão orientar em todo o processo de desenho da aplicação.

Em destaque temos, então, as seguintes guias:

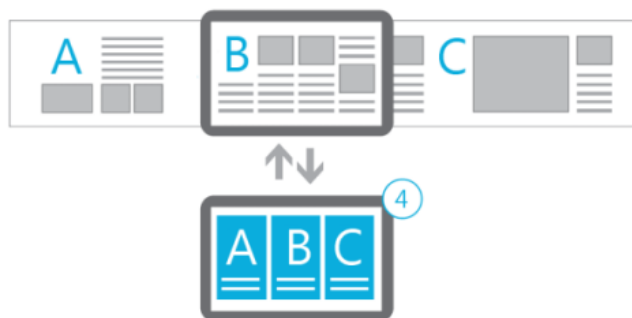
- Navegação
- Commanding (opções)
- Touch
- Advertising (Publicidade)
- Branding (Marca)
- UX Guidelines



Analisemos de uma forma sintetizada os seguintes ecrãs:



1. Cabeçalho e respetivo botão para voltar atrás (normalmente disponível a partir do segundo nível de hierarquia.);
2. Página principal onde é apresentado o conteúdo principal da aplicação;
3. Conteúdo agrupado por categorias de informação ou secções;



4. “Semantic Zoom” sugere que a nossa aplicação tenha a capacidade de ver mais ou menos conteúdo facilitando, assim, o utilizador na navegação e seleção do mesmo.

Relativamente às definições da aplicação, estas devem ser discutidas ao nível do desenho, porque em Windows 8 Store Apps as definições devem ser colocadas no “Settings Charm”, assim como o “Login”/“Logout”.

Expondo um pouco a minha opinião, diria que de toda a informação do Dev Center, os pontos que referi anteriormente são a chave para o sucesso de um bom desenho, e quanto mais próxima a aplicação estiver das recomendações destes guias maior vai ser o sentimento de que estamos perante uma aplicação desenhada para Windows 8. O utilizador vai sentir isso e até mesmo criticar!

Por outro lado, e porque não existe nada perfeito, é compreensível que cada aplicação tenha as suas especificidades e por vezes não é possível seguir estes guias à risca. Neste caso, a melhor forma de percebermos se estamos no caminho certo é termos em conta os requisitos de certificação, porque é isso que faz com que a aplicação seja aprovada ou não.



A sincronização de informação nos vários dispositivos permite enriquecer a aplicação.

Outros aspectos a salientar são:

- Casos de estudo: podemos analisar um exemplo de uma aplicação de iPad “convertida” numa aplicação de Windows 8 Store App, ou mesmo o exemplo de um website;

TEMA DA CAPA

WINDOWS 8 STORE APPS – DO SONHO À REALIDADE

- Inspiração: exemplo de aplicações para diferentes contextos (jogos, compras, desporto, viagem, etc.)

Falemos agora do desenvolvimento. A segunda referência mencionada foi o dev.windows.com, para muitos tida como um ponto de partida.

Nesta referência podemos obter, para além das ferramentas, os SDKs, exemplos, bem como a aquisição da conta para o [Windows Store Dashboard](http://WindowsStoreDashboard.com).

Nota: não confundir licença de desenvolvimento para Windows 8 (a qual é gratuita e permite instalar, desenvolver, testar e avaliar aplicações para Windows 8 Store) com a conta para o [Windows Store Dashboard](http://WindowsStoreDashboard.com) (que permite submeter as aplicações na Store). Neste ponto recomendo a leitura do ["Get developer license"](http://GetDeveloperLicense.com).

Para começarmos a desenvolver a aplicação vamos necessitar do [Visual Studio 2012 Express para Windows 8](http://VisualStudio2012Express.com), que inclui o Windows 8 SDK, e o Blend que permite associar uma aplicação à Store assim como criar o pacote da aplicação. Esta versão é gratuita e permite o desenvolvimento sem ter quaisquer custos adicionais.

A integração do Blend com o Visual Studio vai facilitar o desenvolvimento da aplicação ao nível do desenho da interface com utilizador.

Uma pequena nota: é necessário ter o Windows 8 instalado! Pois não é possível desenvolver Windows 8 Store Apps em Windows 7. No entanto, e apesar de não ser a melhor solução e para o caso de não ser possível uma atualização (imediate) do SO, recomendo o uso do Windows 8 por um período experimental de 90 dias numa máquina virtual ou usando um VHD com boot associado.

Para mais informações sobre esta matéria recomendo a consulta da referência: <http://bit.ly/TZ7zoc>

No dev.windows.com encontramos um conjunto de SDK os quais ajudam no processo de desenvolvimento, onde destaco os seguintes:

- Live SDK: fornece o Identity API (obter perfil do utilizador), Hotmail API, SkyDrive API, Messenger API;

Numa aplicação é possível usar o Windows Live Id, através do Live SDK, como credencial para aceder à aplicação. Isto poderá ser interessante no sentido de que não é necessário criar um sistema de login e recuperação de senha de acesso, o que para o utilizador é menos uma conta que tem que criar!

- Bing Maps SDK for Windows Store apps: acesso à API do Bing Maps;
- Windows 8 Ads in Apps SDK: permite adicionar à aplicação publicidade para através disso obter rendimento;
- Windows Azure Mobile Services: API para integração com Azure;
- Visual Studio Extensions for the Windows Library for JavaScript;

Já temos as ferramentas e os SDKs, então já podemos iniciar o desenvolvimento, mas para isso é preciso decidir que linguagem de programação iremos usar.

Uma das grandes novidades do desenvolvimento de Windows 8 Store Apps é a possibilidade de desenvolver aplicações nativas:

- XAML e C# ou VB
- XAML e C++
- HTML e Javascript

Aumentando o leque de programadores com conhecimentos para desenvolver Windows 8 Store Apps.

No Dev Center encontramos referências para cada linguagem, o que permite uma rápida integração no desenvolvimento. Neste caso recomendo a leitura de:

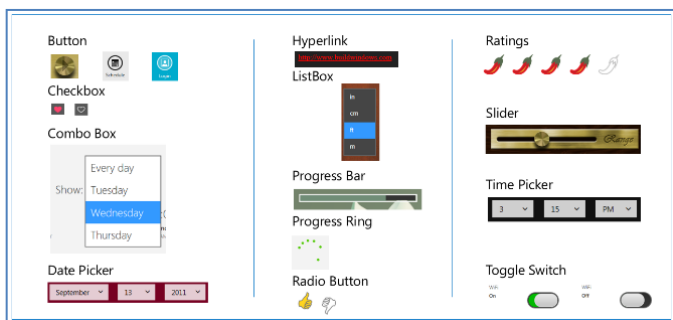
- [EN] [Create your first Windows Store app using JavaScript](http://CreateYourFirstWindowsStoreAppUsingJavaScript.com)
- [EN] [Create your first Windows Store app using C# or Visual Basic](http://CreateYourFirstWindowsStoreAppUsingCSharpOrVisualBasic.com)
- [EN] [Create your first Windows Store app using C++](http://CreateYourFirstWindowsStoreAppUsingCplusplus.com)

Por outro lado, a API oferece um conjunto de controlos, que são possíveis de customizar. Assim, usando o Blend esta customização é facilitada, uma vez que é possível extrair o estilo que é definido por omissão e por sua vez alterá-lo à medida.

Para mais detalhes aconselho a consulta da referência [En] [Blend for Visual Studio 2012 \(Windows Store apps\)](http://BlendForVisualStudio2012.com)

TEMA DA CAPA

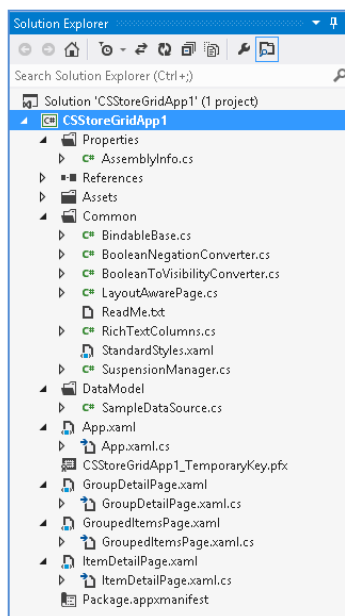
WINDOWS 8 STORE APPS – DO SONHO À REALIDADE



Controlos com estilo customizado

Passando à prática, o Visual Studio 2012 fornece “templates” relevantes, os quais vão facilitar e aumentar a produtividade no desenvolvimento da aplicação. De uma forma geral destaca os seguintes:

- **Blank App:** projeto que contém a estrutura com apenas uma página em branco e sem conteúdo.
- **Grid App:** projeto que contém uma base para uma estrutura que está associada à apresentação de conteúdo por grupos. Exemplos: aplicação de compras, de notícias, leitores de RSS e aplicações de fotos/vídeos.
- **Split App:** projeto que contém uma base de estrutura para uma página principal e uma página de detalhe. Exemplos: aplicação de correio eletrónico e de resultados desportivos.



Projeto de HTML/Javascript usando uma Grid App

Para mais detalhes recomendo leitura da referência: [Visual Studio templates](#).

No Dev Center temos acesso a um conjunto de exemplos através da [MSDN Gallery for Windows Store app samples](#).

Estes exemplos não precisam de ser executados um a um, mas apenas quando é realmente necessário. Isto é, suponhamos que queremos serializar objectos, talvez seja interessante pesquisar por “Serialization” e depois executar os exemplos disponíveis, para uma melhor compreensão do assunto em causa consultar a [API](#).

Com toda esta informação, temos o que é necessário para desenvolver uma aplicação, mas deixo, ainda, mais algumas notas:

- Para os mais experientes em tecnologias, como o WPF, Silverlight e Windows Phone, a integração no desenvolvimento de Windows 8 Store Apps é quase direta, no entanto, chamo atenção que a API em causa é diferente e por essa razão existem funcionalidades de WPF, Silverlight ou Windows Phone que não existem em Windows Store Apps! Outra questão que se vai destacar, é o conceito dos métodos assíncronos usando as palavras-chaves Async/Await, mas isso tem mais a ver com a .Net Framework 4.5;
- Para aqueles que por alguma razão precisam de fazer componentes em várias linguagens recomendo o uso de Windows Runtime Components. Neste caso vejamos um caso prático, vamos supor que é preciso criar uma biblioteca em C++ com funcionalidades que devem ser usadas num projecto de XAML/C#. Se a biblioteca de C++ for um Windows Runtime Component é possível adicionar a referência da biblioteca ao projecto de C# e aceder às classes, e num,... como se de código C# se tratasse;
- Para os programadores de IOS está disponível no Dev Center um conjunto de recursos que os vão ajudar a instalar o Windows 8 e respetivas ferramentas num Mac. Para além disso, vão encontrar diversos conteúdos para que possam integrar-se nos conceitos associados às Windows 8 Store Apps, assim como mapeamento da API e uma comparação entre Objective-C com C#. Aqui é caso para dizer que os programadores de IOS não vão ter desculpa para não fazerem as suas aplicações para Windows 8! Mas para mais detalhes, aconselho a consulta da referência principal [EN] <http://bit.ly/UxvrlQ>.

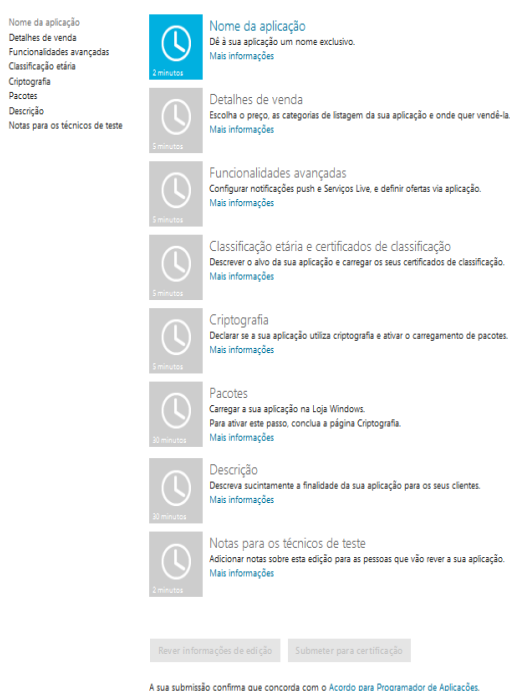
Depois da concessão da aplicação é necessário efetuar testes e validações de acordo com os requerimentos da Store, os quais consta da referência: [EN] <http://bit.ly/T82aeB>, sendo que é um requisito testar a aplicação com o Windows App Cert Kit.

Saliento que estes testes irão fazer uma primeira filtragem de possíveis erros e problemas mais comuns.

No final do teste, feito com o Windows App Cert Kit, é obtido um resultado, ou seja, vamos saber se a nossa aplicação passou ou não passou. Para o caso de não ter passado é preciso analisar o problema e, claro está, corrigi-lo! Chamo a atenção que o teste do WACK não garante que no processo de certificação não sejam encontrados detalhes que possam invalidar a aplicação.

Finalmente, e para tornar o sonho real, isto é, tornar a aplicação pública na Store é necessário submetê-la no Windows Store Dashboard.

Submeter uma aplicação



Ecrã dos passos a completar para submeter uma aplicação na Store

Depois de finalizado todo o processo de submissão, a aplicação irá passar pelo processo de certificação, o qual é constituído por várias etapas. Neste ponto o utilizador será informado sobre quanto tempo demora cada etapa.



Ecrã do estado de certificação

Por fim gostaria ainda de referir que quando a aplicação fica disponível na Store é possível acompanhar a mesma através do Dashboard. Na realidade este painel permite-nos saber o número de vezes que aplicação foi obtida, ver os relatórios com as classificações/comentários, erros que ocorreram na aplicação do lado do utilizador e transações monetárias associadas à aplicação. De salientar que nestes relatórios é possível aplicar filtros por mercado, grupos etários e sexo.

Em conclusão, para desenvolver uma Windows 8 Store App é recomendável a leitura das “UX guidelines”, fazer um pequeno protótipo de forma a perceber como a informação vai ser apresentada e como vai ser o seu fluxo.

Por outro lado, é conveniente ter sempre em mente os principais princípios do Windows 8 Store Apps, como seja:

- Uso do estilo de desenho próprio;
- Rápido e fluido;
- Aplicações escaláveis aos vários ecrãs e com suportes para as várias vistas;
- Uso da Partilha e da Pesquisa;
- Uso de “tiles” secundários e atualizações constantes, assim como guardar o estado da aplicação para que se tenha a perceção que a aplicação está sempre a “correr”;
- E usar “roaming” de definições para manter dados entre dispositivos.

AUTOR



Escrito Por Sara Silva

é mais recente Microsoft MVP em Portugal. De formação base, licenciatura em Matemática – Especialidade em Computação, pela Universidade de Coimbra, é também Microsoft Certified Professional Developer. Atualmente o seu foco de desenvolvimento incide em Windows Phone e Windows 8 Store Apps.

A PROGRAMAR

Threads, Semáforos e Deadlocks – O Jantar dos Filósofos

Programação em OpenCL

Accionamento de Servos Usando Arduino

A web em tempo real na plataforma ASP.NET

Bubble Sort – A Técnica da Bolha

Threads, Semáforos e Deadlocks – O Jantar dos Filósofos

Neste artigo o nosso assunto central será a programação concorrente. Este paradigma foca-se principalmente na interação e execução de multitarefas, assim sendo, uma das principais preocupações quando nos deparamos com um algoritmo deste género são os problemas que podem ocorrer na sincronização e na partilha de recursos.

As vantagens da programação concorrente são o aumento do desempenho do programa, devido ao aumento da quantidade de tarefas que podem ser executadas durante um determinado período de tempo.

Ao longo das próximas linhas vamos debruçarmo-nos sobre alguns factos deste género de programação.

Como referimos há pouco um dos principais problemas a ter em conta é o problema da sincronização. E podemos considerar vários casos neste problema, quer seja com ler e/ou escrever um ficheiro, partilhar recursos ou ter o problema do produtor-consumidor com um entrepósito limitado.

Mas antes de avançarmos, vamos recuar ainda um pouco atrás de modo a que o leitor se recorde e/ou familiarize mais simplesmente com esta temática.

A programação concorrente é um paradigma de programação usado na construção de programas de computador que fazem uso da execução concorrente de várias tarefas computacionais que podem ser implementadas como programas separados ou como um conjunto de threads criadas por um único programa. Nesta altura faz sentido diferenciar o termo processo do termo thread, enquanto que para executar um processo precisamos de uma pilha e de valores de registo que nos dizem o estado desse processo, uma thread é uma linha ou contexto de execução, ou seja é uma forma de um processo se dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

As threads podem partilhar o mesmo espaço de memória assim como podem partilhar as mesmas variáveis, ao contrário dos processos que têm as suas próprias variáveis, pilha e alocação de memória.

Por este motivo caro leitor, é fácil admitirmos que criar e finalizar uma thread é mais rápido e mais simples do que fazer a mesma operação utilizando processos. A comunicação entre threads é também mais simples que a comunicação entre processos, uma vez que as variáveis usadas são comuns.

As threads podem ser implementadas pelo programador ou serem implementados automaticamente pelo Kernel. Por exemplo, normalmente o Windows não escala processos mas sim threads.

Algumas das operações mais usadas no âmbito das threads envolvem a criação de uma thread e a junção de threads (`pthread_create` e `pthread_join`).

Voltando então a uma das perguntas principais: Porque sincronizar?

Porque o acesso concorrente a dados e recursos partilhados pode criar uma situação de inconsistência desses mesmos recursos, isto porque as várias instâncias acedem e manipulam-nos de uma forma “simultânea”, dando azo a situações de inconsistência e de falha. Então para que uma rotina ou programa seja consistente são precisos mecanismos que assegurem a execução ordenada e correta dos processos cooperantes.

Um dos mecanismos computacionais que podemos usar para que não haja inconsistência de recursos são os semáforos. Um semáforo é uma variável, um mecanismo de sincronização sem espera ativa. Esta variável pode ser manipulada através de duas primitivas atómicas, isto é, que não podem ser interrompidas por processos.

A sua função é controlar o acesso a recursos partilhados num ambiente multitarefa. A invenção desse tipo de variável é atribuída a Edsger Dijkstra.

Quando declaramos um semáforo, o seu valor indica quantos processos (ou threads) podem ter acesso ao recurso partilhado. Assim, para partilhar um recurso o leitor deve verificar qual o valor do semáforo para saber qual a ação a executar.

Assim e de forma a consolidar os conhecimentos, as principais operações sobre semáforos são:

Inicialização: Recebe como parâmetro um valor inteiro indicando a quantidade de processos que podem aceder a um determinado recurso.

Operação wait: Decrementa o valor do semáforo. Se o semáforo está com valor zero, o processo é posto em espera. Para receber um sinal, um processo executa `WAIT` e bloqueia se o Semáforo impedir a continuação da sua execução.

Operação signal: Se o semáforo estiver com o valor zero e existir algum processo em espera, um processo será acordado. Caso contrário, o valor do semáforo é incrementado. Para enviar um sinal, um processo executa um `signal`.

As operações de incrementar e decrementar devem ser ope-

A PROGRAMAR

THREADS, SEMÁFOROS E DEADLOCKS

rações atômicas, ou indivisíveis, isto é, a instrução deve ser executada na totalidade sem que haja interrupções. Enquanto um processo estiver a executar uma dessas duas operações, nenhum outro processo pode executar outra operação nesse mesmo semáforo, devendo esperar que o primeiro processo encerre a sua operação. Essa obrigação evita condições de disputa entre as várias threads.

Para evitar espera, o que desperdiça processamento da máquina, a operação Wait utiliza uma estrutura de dados (geralmente uma FIFO). Quando um processo executa essa operação e o semáforo tem o seu valor a zero, este valor é posto na estrutura. Quando um outro processo executar a operação Signal havendo processos nessa mesma estrutura, uma delas é retirada (no caso FIFO, é retirada a primeira que entrou).

Tendo em conta que há necessidade de garantir a consistência dos recursos, a utilização mais simples do semáforo é em situações na qual se utiliza o princípio da exclusão mútua, isto é, que só um processo é executado de cada vez. Para isso utiliza-se um semáforo binário, com inicialização em 1. Esse semáforo binário atua como um mutex.

O princípio da Exclusão mútua ou mutex, é uma técnica usada em programação concorrente para evitar que dois processos ou threads tenham acesso simultaneamente a um recurso partilhado. Esta partilha dá origem à chamada secção crítica que pode ser protegida da seguinte forma:

```
Pthread_mutex_lock()
```

```
[secção crítica]
```

```
Pthread_mutex_unlock()
```

Então, um meio simples para exclusão mútua é a utilização do tal semáforo binário, isto é, que só pode assumir dois valores distintos, 0 e 1. O fecho do semáforo deve ser feito antes de utilizar o recurso e deve ser libertado só depois do uso o recurso. Tendo em conta que apenas um processo pode entrar de cada vez na secção crítica, então os outros processos devem esperar pela sua vez.

Contudo o uso do princípio da exclusão mutua, pode originar alguns problemas, como por exemplo deadlocks ou starvation (inanição).

Um deadlock, ou bloqueio, é referente a uma situação em que existe um impasse entre dois ou mais processos, isto é, no nosso contexto, quando dois ou mais processos tentam aceder ao mesmo recurso, bloqueiam-se mutuamente impedindo a boa continuação do algoritmo.

Ocorre quando várias threads do conjunto estão à espera da libertação de um recurso por uma outra thread que, por sua vez, aguarda a libertação de outro recurso alocado ou de-

pendente da primeira thread.

A starvation por seu lado, ocorre quando uma thread nunca é executada por estar à espera de uma resposta que uma outra thread supostamente lhe dará, contudo esta última thread por algum motivo (a thread pode morrer ou ficar bloqueada) nunca lhe dá a resposta esperada. Diz-se que neste caso a thread se torna um zombie, esperando uma resposta que possivelmente nunca terá.

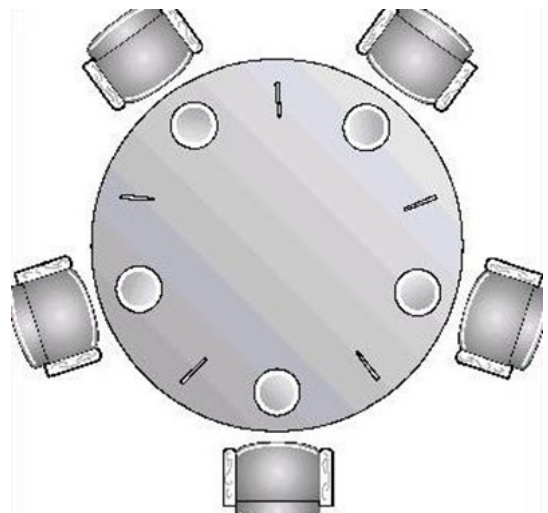
Quando o escalonamento não é feito adequadamente, pode haver inanição. Uma solução para esta situação é a existência de um tempo máximo de espera.

A ocorrência da inanição dá-se também quando os programas entram em ciclo infinito (razão pela qual também se dá o nome de preterição indefinida a esta situação) e não fazem nenhum progresso no processamento, ao contrário do deadlock, que ocorre quando os processos permanecem bloqueados, dependendo da liberação dos recursos por eles alocados.

Após esta pequena explicação teórica, vamos caro leitor voltar ao algoritmo escolhido para fonte central deste artigo, o jantar dos filósofos. Este é considerado um problema clássico de programação concorrente, criado em 1965 por Edsger Dijkstra e tem como objetivo principal demonstrar a alocação de recursos quando temos processos que concorrem pela alocação dos mesmos, ou seja, os recursos disponíveis são escassos. Neste problema há a necessidade de reservar recursos entre vários processos sem deadlock ou starvation.

Algoritmo – O Jantar dos Filósofos

Considere 5 filósofos que passam a vida a pensar e a comer. Partilham uma mesa redonda rodeada por 5 cadeiras sendo que cada uma das cadeiras pertence a um filósofo. No centro da mesa encontra-se uma taça de arroz e estão 5 garfos na mesa, um para cada filósofo.



A PROGRAMAR

THREADS, SEMÁFOROS E DEADLOCKS

Quando um filósofo pensa não interage com os seus colegas. De tempos em tempos, cada filósofo fica com fome e tenta apanhar os dois garfos que estão mais próximos (os garfos que estão ou à esquerda ou à direita). O filósofo apenas pode apanhar um garfo de cada vez e como o leitor compreende, não pode apanhar um garfo se este estiver na mão do vizinho. Quando um filósofo esfomeado tiver 2 garfos ao mesmo tempo ele come sem largar os garfos. Apenas quando acaba de comer, o filósofo pousa os garfos, libertando-os e começa a pensar de novo. O nosso objetivo é ter uma representação/implementação que nos permita simular este jantar sem que haja problemas de deadlock ou starvation.

Para isso, o jantar será modelado usando uma thread para representar cada filósofo e usaremos semáforos para representar cada garfo. Quando um filósofo tenta agarrar um garfo executando uma operação wait no semáforo, quando o filósofo larga o garfo executa uma operação Signal nesse mesmo semáforo. Cada filósofo (thread) vai seguir o algoritmo, ou seja, todos fazem as mesmas ações. Como deve estar já o leitor a pensar, o facto de seguirem o mesmo algoritmo pode dar azo à situação de deadlock, daí a utilização das primitivas de sincronização WAIT e SIGNAL. Uma outra possibilidade de deadlock seria o facto de mais do que um filósofo ficar com fome ao mesmo tempo, os filósofos famintos tentariam agarrar os garfos ao mesmo tempo. Isto é outro ponto que uma solução satisfatória terá que ter em atenção, deve ser salvaguardado o facto de um filósofo não morrer à fome. Devemos recordar o leitor que uma solução livre de deadlock não elimina necessariamente a possibilidade de um filósofo morrer esfomeado.

Por todos estes motivos, o jantar dos filósofos é um algoritmo que deve ser implementado com algum cuidado por parte do programador.

Com o recurso à rede das redes, o leitor pode facilmente encontrar vários exemplos de implementação deste algoritmo em várias linguagens, contudo, também este artigo seguidamente apresenta um exemplo de uma implementação simples em C e em Java para que o leitor possa comparar os respetivos códigos.

the PHILOSOPHER'S PARADOX



what am i
gonna do now?

Como criar Threads em Java (Exemplo):

Criamos uma classe que implemente a interface Runnable:

```
public class OlaRunnable implements Runnable
{
    public void run()
    {
        System.out.println("A Thread começou!");
    }

    public static void main(String args[])
    {
        (new Thread(new OlaRunnable())).start();
    }
}
```

ou então criamos uma subclasse de Thread:

```
public class OlaThread extends Thread
{
    public void run()
    {
        System.out.println("A Thread começou!");
    }

    public static void main(String args[])
    {
        (new OlaThread()).start();
    }
}
```

É necessário definir o método run(), requerido pela interface Runnable, para qualquer uma das opções escolhidas pelo programador. Esse método define o que irá acontecer após invocar o método start() na instância previamente criada.

Interromper uma Thread:

Para pararmos a execução de uma Thread, invocamos o método Thread.sleep() que recebe como parâmetro, o número de milissegundos que a Thread irá ficar bloqueada.

Este método pode lançar uma exceção (InterruptedException), logo é necessário "apanhá-la" de forma correcta.

Semáforos:

Os semáforos são uma primitiva de concorrência muito básica que suportam o acesso concorrente a zonas de memória partilhada, ainda que alguns problemas de sincronização possam surgir, tais como deadlocks e starvation. Para que esses problemas possam ser eliminados, convém usar soluções melhores para exclusão mútua, como por exemplo, monitores.

Segue um exemplo de como utilizar um semáforo em Java:

```
import java.util.concurrent.Semaphore;
public class Semaforo
{
    //(...)
```

A PROGRAMAR

THREADS, SEMÁFOROS E DEADLOCKS

```
Semaphore s = new Semaphore(1);
s.acquire();
// Código crítico
s.release();
//(...)
}
```

Como criar Threads em C (Exemplo):

```
#include <stdio.h>
#include <pthread.h>

void *hello()
{
    printf("Hello World\n");
    pthread_exit(NULL);
}

int main()
{
    pthread_t id;
    int status;
    status = pthread_create(&id , NULL , hello ,
                           NULL);

    if(status!=0)
        exit(-1);
    pthread_exit(NULL);
}
```

Para compilar e executar este programa com o gcc, o leitor deve fornecer os

Commandos:

```
gcc -o hellothread hellothread.c -lpthread
./hellothread
```

Semáforos em C:

Utilizam a biblioteca <semaphore.h>

Tipo: sem_t representação dum semáforo

- int sem_init(sem_t *sem, int pshared, unsigned int value);
- int sem_wait(sem_t * sem);
- int sem_post(sem_t * sem);

Jantar dos Filósofos – Implantação em Java:

Classe Jantar:

```
public class Jantar
{
    public static void main (String[] args)
    {
        Mesa mesa = new Mesa ();
        for (int filosofo = 0; filosofo < 5;
```

```
        filosofo++)
        {
            new Filosofo("Filosofo_" +
                filosofo, mesa, filosofo).start();
        }
    }
}
```

Classe Filosofo:

```
public class Filosofo extends Thread
{
    final static int TEMPO_MAXIMO = 100;
    Mesa mesa;
    int filosofo;
    public Filosofo (String nome, Mesa mesadejantar,
                    int fil)
    {
        super(nome);
        mesa = mesadejantar;
        filosofo = fil;
    }
    public void run ()
    {
        int tempo = 0;
        while (true)
        {
            tempo = (int)
                (Math.random() * TEMPO_MAXIMO);
            pensar(tempo);
            getGarfos();
            tempo = (int)
                (Math.random() * TEMPO_MAXIMO);
            comer(tempo);
            returnGarfos();
        }
    }
    public void pensar (int tempo)
    {
        try
        {
            sleep(tempo);
        }
        catch (InterruptedException e)
        {
            System.out.println("O Filósofo pensou em
                demasia");
        }
    }
    public void comer (int tempo)
    {
        try
        {
            sleep(tempo);
        }
        catch (InterruptedException e)
        {
            System.out.println("O Filósofo comeu em
                demasia");
        }
    }
    public void getGarfos()
    {
        mesa.pegarGarfos(filosofo);
    }
    public void returnGarfos()
    {
        mesa.returningGarfos(filosofo);
    }
}
```


A PROGRAMAR

THREADS, SEMÁFOROS E DEADLOCKS

Classe Mesa:

```
public class Mesa
{
    final static int PENSANDO = 1;
    final static int COMENDO = 2;
    final static int FOME = 3;
    final static int NR_FILOSOFOS = 5;
    final static int PRIMEIRO_FILOSOFO = 0;
    final static int ULTIMO_FILOSOFO =
        NR_FILOSOFOS - 1;
    boolean[] garfos = new boolean[NR_FILOSOFOS];
    int[] filosofos = new int[NR_FILOSOFOS];
    int[] tentativas = new int[NR_FILOSOFOS];
    public Mesa()
    {
        for (int i = 0; i < 5; i++)
        {
            garfos[i] = true;
            filosofos[i] = PENSANDO;
            tentativas[i] = 0;
        }
    }
    public synchronized void pegarGarfos (int filosofo)
    {
        filosofos[filosofo] = FOME;
        while (filosofos[aEsquerda(filosofo)] ==
            COMENDO || filosofos[aDireita(filosofo)]
                == COMENDO)
        {
            try
            {
                tentativas[filosofo]++;
                wait();
            }
            catch (InterruptedException e)
            {
                System.out.println("O Filósofo
                    morreu devido a starvation");
            }
        }
        tentativas[filosofo] = 0;
        garfos[garfoEsquerdo(filosofo)] = false;
        garfos[garfoDireito(filosofo)] = false;
        filosofos[filosofo] = COMENDO;
        imprimeEstadosFilosofos();
        imprimeGarfos();
        imprimeTentativas();
    }
    public synchronized void returningGarfos (int filosofo)
    {
        garfos[garfoEsquerdo(filosofo)] = true;
        garfos[garfoDireito(filosofo)] = true;
        if (filosofos[aEsquerda(filosofo)] == FOME
            || filosofos[aDireita(filosofo)] == FOME)
        {
            notifyAll();
        }
        filosofos[filosofo] = PENSANDO;
        imprimeEstadosFilosofos();
        imprimeGarfos();
        imprimeTentativas();
    }
    public int aDireita (int filosofo)
    {
        int direito;
        if (filosofo == ULTIMO_FILOSOFO)
        {
            direito = PRIMEIRO_FILOSOFO;
        }
        else
        {
            direito = filosofo + 1;
        }
    }
}
```

```
    }
    return direito;
}
public int aEsquerda (int filosofo)
{
    int esquerdo;
    if (filosofo == PRIMEIRO_FILOSOFO)
    {
        esquerdo = ULTIMO_FILOSOFO;
    }
    else
    {
        esquerdo = filosofo - 1;
    }
    return esquerdo;
}
public int garfoEsquerdo (int filosofo)
{
    int garfoEsquerdo = filosofo;
    return garfoEsquerdo;
}
public int garfoDireito (int filosofo)
{
    int garfoDireito;
    if (filosofo == ULTIMO_FILOSOFO)
    {
        garfoDireito = 0;
    }
    else
    {
        garfoDireito = filosofo + 1;
    }
    return garfoDireito;
}
public void imprimeEstadosFilosofos ()
{
    String texto = "*";
    System.out.print("Filósofos = [ ");
    for (int i = 0; i < NR_FILOSOFOS; i++)
    {
        switch (filosofos[i])
        {
            case PENSANDO :
                texto = "PENSANDO";
                break;

            case FOME :
                texto = "FOME";
                break;

            case COMENDO :
                texto = "COMENDO";
                break;
        }
        System.out.print(texto + " ");
    }
    System.out.println("]");
}
public void imprimeGarfos ()
{
    String garfo = "*";
    System.out.print("Garfos = [ ");
    for (int i = 0; i < NR_FILOSOFOS; i++)
    {
        if (garfos[i])
        {
            garfo = "LIVRE";
        }
        else
        {
            garfo = "OCUPADO";
        }
        System.out.print(garfo + " ");
    }
}
```

A PROGRAMAR

THREADS, SEMÁFOROS E DEADLOCKS

```
System.out.println("");
}
public void imprimeTentativas ()
{
    System.out.print("Tentou comer = [ ");
    for (int i = 0; i < NR_FILOSOFOS; i++)
    {
        System.out.print(filosofos[i] + " ");
    }
    System.out.println("");
}
}
```

“**As vantagens da programação concorrente são o aumento do desempenho do programa, devido ao aumento da quantidade de tarefas que podem ser executadas durante um determinado período de tempo**”

Jantar dos Filósofos – Implantação em C:

```
#include<stdio.h>
#include<stdlib.h>
#include<semaphore.h>
#include<pthread.h>

#define N 5
#define PENSAR 0
#define FOME 1
#define COMER 2
#define ESQUERDA (nfilosofo+4)%N //agarrar garfo
//da esquerda
#define DIREITA (nfilosofo+1)%N //agarrar garfo
//da direita

void *filosofo(void *num);
void agarraGarfo(int);
void deixaGarfo(int);
void testar(int);
```

```
sem_t mutex;
sem_t S[N]; //inicializacao do semáforo

int estado[N];
int nfilosofo[N]={0,1,2,3,4};

void *filosofo(void *num)
{
    while(1)
    {
        int *i = num;
        sleep(1);
        agarraGarfo(*i);
        sleep(1);
        deixaGarfo(*i);
    }
}

void agarraGarfo(int nfilosofo)
{
    sem_wait(&mutex);
    estado[nfilosofo] = FOME;
    printf("Filosofo %d tem fome.\n",nfilosofo+1);
    //+1 para imprimir filosofo 1 e nao filosofo 0
    testar(nfilosofo);
    sem_post(&mutex);
    sem_wait(&S[nfilosofo]);
    sleep(1);
}

void deixaGarfo(int nfilosofo)
{
    sem_wait(&mutex);
    estado[nfilosofo]=PENSAR;
    printf("Filosofo %d deixou os garfos %d e %d.\n",nfilosofo+1,ESQUERDA+1,nfilosofo+1);

    printf("Filosofo %d esta a pensar.\n",nfilosofo+1);

    testar(ESQUERDA);
    testar(DIREITA);
    sem_post(&mutex);
}

void testar(int nfilosofo)
{
    if(estado[nfilosofo]==FOME && estado[ESQUERDA]
        !=COMER && estado[DIREITA]!=COMER)
    {
        estado[nfilosofo]=COMER;
        sleep(2);
        printf("Filosofo %d agarrou os garfos %d e %d.\n",nfilosofo+1,ESQUERDA+1,nfilosofo+1);
        printf("Filosofo %d esta a comer.\n",nfilosofo+1);
        sem_post(&S[nfilosofo]);
    }
}

int main()
{
    int i;
    pthread_t thread_id[N]; //identificadores das
    //threads

    sem_init(&mutex,0,1);
    for(i=0;i<N;i++)
        sem_init(&S[i],0,0);
    for(i=0;i<N;i++)
    {
        pthread_create(&thread_id
            [i],NULL,filosofo,&nfilosofo[i]);
        //criar as threads
        printf("Filosofo %d esta a pensar.\n",i+1);
    }
}
```

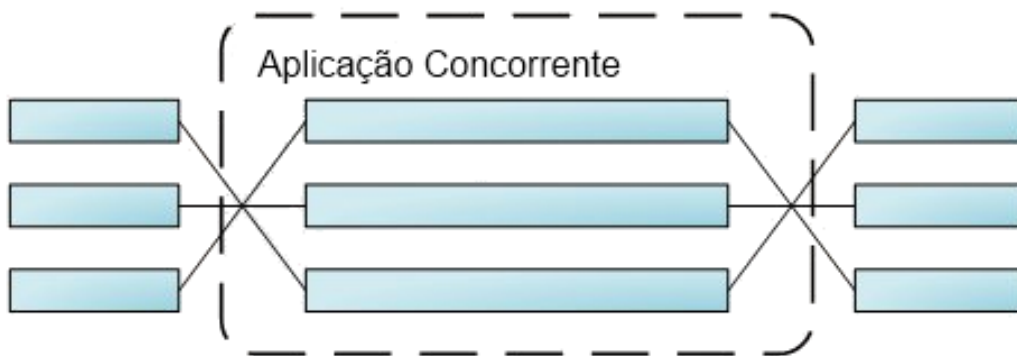
```
for(i=0; i<N; i++)  
    pthread_join(thread_id[i], NULL); //para  
//fazer a junção das threads  
return(0);  
}
```

“ **A programação concorrente é um paradigma de programação usado na construção de programas que fazem uso da execução concorrente de várias tarefas** ”

Chegados à reta final deste artigo gostaríamos mais uma vez de recordar o caro leitor da importância do bom uso dos mecanismos de sincronização.

Apesar de o jantar dos filósofos ser um problema clássico de sincronismo, é também uma situação teórica, contudo a verdade é que no “mundo real” há várias situações em que mecanismos de sincronização são usados e extremamente necessários, quer seja por exemplo, por uma simples transação bancária (em que há a necessidade de garantir a integridade e a boa sincronização do saldo real da nossa conta) ou até mesmo o algoritmo de sincronização dos sinais luminosos de trânsito (em que para que não haja acidentes há que garantir que duas ruas com vias contrárias, que terminem no mesmo cruzamento, não permitam a passagem de trânsito ao mesmo tempo - o cruzamento seria neste caso a secção crítica).

Este artigo é o primeiro de uma série de artigos de programação tendo como base principal a Linguagem C e a Linguagem Java como base auxiliar, que esperamos que siga atentamente.



AUTOR



Escrito por Rita Peres

Natural de Castelo Branco e estudante de Engenharia Informática da Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010 (<https://www.facebook.com/rita.aperes>)



Escrito por Diogo Barros

Atualmente a frequentar, como finalista da licenciatura de Ciências da Computação da Universidade do Minho, Diogo Barros sempre demonstrou interesse pela programação e novas tecnologias. Tem conhecimentos sólidos de Haskell, Java, C, PHP e Erlang. Tem, como passatempo, a prática de ping-pong e musculação, além de produção de Uplifting Trance e DJing. Username no Fórum [P@P](#): BlueKore

25 DE MAIO DE 2013 @ LISBOA

PROGRAMAR 2013

O EVENTO PRESENCIAL DA COMUNIDADE

PORTUGAL-A-PROGRAMAR

Programação em OpenCL

A norma OpenCL (*Open Computing Language*) foi impulsionada pela empresa Apple com o intuito de uniformizar a programação para dispositivos de processamento de alto desempenho que suportem execução paralela. Assim, o OpenCL tem por alvo dispositivos de processamento, sejam eles processadores *multicores*, placas gráficas com capacidade de processamento paralelo (GPU - *Graphical Processing Unit*), ou outros dispositivos de co-processamento como processadores de sinais (vulgarmente designados por DSP, *Digital Signal Processors*). Presentemente, a norma OpenCL está sob o controlo do Khronos Group (<http://www.khronos.org/>), um consórcio que coordena normas abertas tais como a bem conhecida OpenGL (norma de programação gráfica), a variante OpenGL/ES (OpenGL para sistemas embarcados) e ainda a norma OpenAS/ES (aceleração de áudio para sistemas embarcados).

A norma OpenCL foi adotada por um vasto conjunto de grandes empresas da área da informática (AMD/ATI, NVIDIA, Intel, IBM, Samsung, Google, entre outras). Deste modo, ao contrário da plataforma CUDA (ver “Introdução à Programação em CUDA”, revista Programar nº38, Dezembro 2012) que é uma plataforma proprietária que está apenas disponível para GPUs da NVIDIA, a norma OpenCL é suportada numa vasta gama de dispositivos desde as GPUs da AMD/ATI e da NVIDIA, ao processador Cell e ainda nas mais recentes placas gráficas do tipo HD e CPUs *multicore* da Intel. Contudo, o suporte varia, sendo ainda poucos os dispositivos que disponibilizam a versão 1.2 da norma OpenCL publicada em Novembro de 2011 (Khronos Group OpenCL Working Group, 2012). Neste artigo introduzem-se os conceitos básicos de programação paralela na norma OpenCL, focando somente o OpenCL na perspetiva de dispositivos GPUs, dado ser esse o cenário mais comum.

O objetivo do OpenCL é o de permitir explorar as capacidades de execução em paralelo dos dispositivos de processamento. Para tal, a norma OpenCL define uma API (*Application Programming Interface*) para a linguagem C complementada com extensões para paralelismo. Para além do suporte direto para a linguagem C, existe ainda suporte para OpenCL noutras linguagens de programação como o C++, o Python (PyOpenCL) e o Java (Jocl.org), entre outras.

Modelo de interação

À semelhança do paradigma CUDA, o modelo de interação da norma OpenCL assenta num programa executado no sistema hospedeiro (CPU) que lança operações para execução paralela no(s) dispositivo(s) OpenCL através daquilo que se designa por *kernel*. Um kernel OpenCL assume um for-

mato similar a uma função escrita em linguagem C acrescido de alguns modificadores específicos do OpenCL. A Listagem 1 mostra o código do kernel OpenCL **Inverte1D** cujo propósito é o de inverter uma imagem no formato PGM (Portable Gray Map). O PGM é um formato para a representação de imagens a preto e branco (mais adiante, será abordado em maior detalhe o formato PGM). A operação de inversão consiste em construir o negativo da imagem substituindo a intensidade P de cada pixel da imagem pelo inverso, isto é, 255-P. Deste modo, um píxel branco (valor 255) passa a preto (valor 0) e vice-versa.

```
__kernel void Inverte1D(__global const char *
                        inputImg,
                        __global char * outputImg, int largura, int
                        altura)
{
    int NumPixels = largura*altura;
    unsigned int TotalThreads = get_global_size(0);
    unsigned int i = get_global_id(0);
    while(i<NumPixels){
        outputImg[i] = 255-inputImg[i];
        i += TotalThreads;
    }
}
```

Listagem 1: código do kernel Inverte1D

O código do kernel Inverte1D inicia-se pelo modificador **__kernel**. Os parâmetros ponteiros para **char inputImg** e **outputImg** são precedidos pelos modificadores **__global**, indicando que correspondem a endereços de memória do dispositivo OpenCL. O parâmetro **inputImg** corresponde ao vetor de caracteres que contém a imagem a processar, sendo que cada carácter corresponde a um píxel. Por sua vez, o parâmetro **outputImg** corresponde ao espaço de memória do dispositivo onde é escrita a versão invertida da imagem. Finalmente, os parâmetros do tipo inteiro **largura** e **altura** indicam, em píxeis, as dimensões da imagem.

Geometria de execução e coordenadas OpenCL

Em termos práticos, o programador especifica através do processo hospedeiro a execução de um kernel, indicando para o efeito uma determinada geometria de execução. O processo hospedeiro é a instância do programa que corre no CPU da máquina e que interage com o(s) dispositivo(s) OpenCL. Uma geometria de execução define o número de entidades que executam, em paralelo, o código do kernel. Na terminologia OpenCL, as entidades são designadas por *work-items*, e correspondem às *threads* da plataforma CUDA. Os *work-items* são especificados através de uma geometria que pode comportar até três dimensões, designadas por dimensão 0, 1 e 2 e que correspondem à dimensão X, Y

A PROGRAMAR

PROGRAMAÇÃO EM OPENCL

e Z, respetivamente.

Tal como sucede com as threads do CUDA, os *work-items* do OpenCL organizam-se em grupos, designados por *work-groups*. Os work-groups podem eles também apresentar uma geometria com dimensões.

Cada work-item é identificado na geometria de execução através das respetivas coordenadas. Assim, cada work-item é univocamente identificado por um índice global do tipo inteiro, acessível através da função OpenCL **get_global_id(int dim)**, com o parâmetro *dim* a indicar a dimensão pretendida. Uma coordenada apresenta um número de dimensões idêntico ao especificado para a geometria de execução. Assim, no caso de um geometria de execução bidimensional (dimensões 0 e 1), será necessário chamar **get_global_id(0)** e **get_global_id(1)** para obter, respetivamente, o índice global da dimensão 0 e da dimensão 1. O OpenCL disponibiliza ainda o acesso a uma coordenada local através da função **get_local_id(int dim)**. A coordenada local identifica o work-item no contexto do work-group em que se insere. Acresce-se que o próprio work-group é univocamente identificado através da função **get_group_id(int dim)**. Finalmente, as dimensões da geometria de execução podem ser obtidas através de **get_global_size(int dim)** que devolve o número de work-items existentes na dimensão *dim* e **get_local_size(int dim)** que devolve o número de work-items na dimensão *dim* do work-group a que pertence o work-item que chama a função.

get_global_id(int dim)	devolve identificador global
get_local_id(int dim)	devolve identificador local
get_group_id(int dim)	devolve identificador do grupo
get_global_size(int dim)	devolve total de work-items
get_local_size(int dim)	devolve número de work-items do work-group

Tabela 1: funções de identificação e mensuração acessíveis a partir de um kernel OpenCL

Modelo de execução OpenCL

O modelo de execução do OpenCL assenta no facto que os work-items executam, simultaneamente, as mesmas operações sobre dados diferentes, atuando em modo *lock-step*. Por exemplo, no tratamento de uma imagem (Listagem 1), cada work-item está encarregue de processar um píxel. Dado o dispositivo OpenCL executar vários work-items simultaneamente, obtém-se uma execução em paralelo. Assim, no caso de uma execução com 32 work-items e 64 work-groups, a execução poderá, potencialmente, ser efetuada simultaneamente por 2048 entidades (32 x 64). Importa notar que fatores como o acesso à memória e o próprio dispositivo OpenCL (número de *cores*, número e tipo de unidades de computação por *cores*, etc.) limitam a execução impedindo muitas vezes o pleno paralelismo.

Fluxograma de um programa OpenCL

O fluxograma de um programa OpenCL é bastante simples. Primeiro procede-se à alocação da memória no dispositivo OpenCL, dado que o código de um kernel apenas pode aceder à memória do dispositivo (por exemplo, a memória de uma placa gráfica empregue como dispositivo OpenCL). É habitual ser alocado espaço em memória para os dados de entrada e para a escrita dos resultados por parte do kernel. Seguidamente, procede-se à cópia dos dados de entrada da memória do processo hospedeiro para a memória previamente alocada no dispositivo OpenCL. Copiados os dados de entrada, procede-se ao lançamento da execução do kernel, indicando-se a geometria de execução (geometria dos work-items e dos work-groups), bem como os parâmetros a serem passados ao kernel. Terminada a execução do kernel, copiam-se os resultados da memória do dispositivo OpenCL para a memória do programa que corre na máquina hospedeira. Finalmente, procede-se à libertação dos recursos previamente alocados.

Apesar do fluxograma de um programa OpenCL ser simples, o facto do OpenCL suportar uma grande variedade de dispositivos de execução paralela, leva a que a API seja relativamente verbosa, existindo um elevado número de funções a chamar, ao que se acresce o facto de muitas dessas funções comportarem um elevado número de parâmetros. Deste modo, e comparativamente ao CUDA (modo runtime), para um mesmo efeito, um programa OpenCL comporta um maior número de linhas e de chamadas à respetiva API.

Exemplo OpenCL – Inversão de imagem PGM

Para exemplificar a programação na norma OpenCL, este artigo recorre à implementação da operação de inversão de uma imagem PGM. O kernel **Inverte1D** que cria o negativo de uma imagem já foi apresentado na Listagem 1. A designação 1D deve-se ao facto do kernel apenas estar preparado para execução numa geometria unidimensional, recorrendo somente à dimensão 0.

O kernel **Inverte1D** funciona de forma bastante simples. Cada work-item identifica o píxel que tem que processar através da coordenada global devolvida pela chamada a **get_global_id(0)**. Assim, o work-item 0 processa o píxel 0, o work-item 1 processa o píxel 1 e assim sucessivamente. No código do kernel **Inverte1D**, cada work-item mantém o índice do píxel a processar na variável *i*. Por forma a processar imagens que apresentem um número de píxeis superior ao número total de work-items, o kernel assenta num ciclo *while*, com cada work-item a processar, se necessário, mais do que um píxel. Assim, após processar um píxel, o work-item determina o índice do próximo píxel que deve processar, somando à variável *i*, o número total de work-items (variável **TotalThreads**, ela própria preenchida pela função **get_global_size(0)**). Caso o novo valor da variável *i* ainda seja inferior ao total de píxeis da imagem, o work-item efetua

uma nova iteração do ciclo *while*, processando o píxel de índice *i*, e assim sucessivamente.

Formato PGM

O formato de imagem PGM guarda uma imagem em tons de cinza (vulgo preto-e-branco) registando a intensidade do nível de cinzento de cada píxel através de um valor inteiro compreendido entre 0 (píxel preto) e um valor máximo (píxel branco). A Listagem 2 mostra as primeiras linhas do ficheiro PGM *Lena*, imagem usada nos exemplos deste artigo e empregue como referência em muitas áreas de processamento de imagem (Wikipedia). A primeira linha indica que é um ficheiro do tipo P2, isto é, um ficheiro PGM que usa o formato de texto para registar a intensidade de cada píxel da imagem. A segunda linha, iniciada pelo símbolo #, corresponde a uma linha de comentário. Na terceira linha encontram-se as dimensões da imagem: largura seguida da altura, neste caso, 512 para cada dimensão, significando pois que a imagem é quadrada. A quarta linha indica o valor máximo para a intensidade de um píxel que neste caso é de 245. Neste artigo apenas consideramos imagens que tenham um valor máximo menor ou igual que 255, por forma a que a intensidade de cada píxel apenas ocupe um byte, podendo ser guardada por um elemento do tipo *char*. Finalmente, as linhas seguintes representam os píxeis da imagem, num varrimento da esquerda para a direita e de cima para baixo. Dado a imagem apresentar as dimensões 512 por 512, existem 262144 píxeis e consequentemente o ficheiro PGM contém 262144 valores inteiros que descrevem a intensidade de cinzento de cada píxel da imagem. As funções *GravarPGM* e *LerPGM*, disponíveis no código que acompanha este artigo, tratam respetivamente da escrita e da leitura de imagens do tipo PGM.

```
P2
# lena.pgm
512 512
245
162 162 162 161 162 156 163 160 164
(...)
```

Listagem 2: Primeiras linhas do ficheiro lena.pgm



Figura 1: imagem de teste "lena.pgm"

Tratamento de erro no OpenCL

O tratamento de erro é de particular importância no OpenCL, não só pelo facto de muitas operações depender de recursos do sistema mas também devido ao elevado número de funções da API OpenCL necessárias a um programa OpenCL, sendo

que muitas dessas funções comportam elas próprias um elevado número de parâmetros. Todos esses fatores aumentam a probabilidade de erros.

A norma OpenCL apresenta duas formas diferentes para reportar erros. A maior parte das funções retorna um código de estado do tipo *cl_int_err* que indica o sucesso da operação caso tenha o valor de *CL_SUCCESS*. Contudo, existem funções OpenCL que retornam tipos de dados específicos, como por exemplo, *clCreateContext* que devolve um contexto na forma do tipo de dado *cl_context*. Para essas funções, o OpenCL define que o último parâmetro deve conter o endereço de uma variável do tipo *cl_int* que é preenchida pelo OpenCL com o código numérico do eventual erro que possa ter ocorrido. Em qualquer dos casos – funções OpenCL que retornam diretamente um código de erro e funções OpenCL que recorrem ao último parâmetro – o código de erro é descrito por uma constante do préprocessador definido pela norma OpenCL. Por exemplo, caso não seja encontrado um dispositivo OpenCL, será retornado o valor *CL_DEVICE_NOT_FOUND*. Dado que um código numérico é pouco ou nada informativo, é frequente o uso de uma função que recebendo um código de erro, devolve uma representação textual do código de erro. É esse o papel da função *ErroCLToStr*, parcialmente mostrada na Listagem 3 e que se baseia no código da função *sclPrintErrorFlag* que integra o projeto de código aberto *simple-opencl* (*SimpleOpenCL*).

```
char *ErroCLtoStr(cl_int errorCode)
{
    switch(errorCode)
    {
        case CL_SUCCESS:
            return "CL_SUCCESS";
        case CL_DEVICE_NOT_FOUND:
            return "CL_DEVICE_NOT_FOUND";
            (...)
        default:
            return "Unknow code";
    }
}
```

Listagem 3: Listagem parcial da função ErroCLToStr

Para além do uso da função de conveniência *ErroCLToStr*, o tratamento de erro empregue neste artigo assenta em duas macros: *CL_CHECK* e *CL_CHECK_ERR* (Listagem 4). Ambas foram adaptadas das macros disponibilizadas no código de Clifford Wolf (Wolf, 2009). A macro *CL_CHECK* destina-se a funções OpenCL que retornam diretamente o código de erro, ao passo que a macro *CL_CHECK_ERR* deve ser empregue na chamada das funções OpenCL que fazem uso do último parâmetro para o assinalar de eventuais erros, sendo que nesse caso a macro requer que seja especificado o endereço do identificador *_err*, isto é, *&_err* (*_err* é uma variável declarada pela macro) como último parâmetro. Ambas as macros são empregues da mesma forma, especificando-se como parâmetro da macro a chamada à função cujos even-

A PROGRAMAR

PROGRAMAÇÃO EM OPENCL

tuais erros se pretendem tratar. Para além de uniformizar o tratamento de erros, as macros `CL_CHECK` e `CL_CHECK_ERR` minimizam as interferências na legibilidade do código.

```
/** Adaptado de Clifford Wolf
<clifford@clifford.at>
 * http://svn.clifford.at/tools/trunk/examples/
cldemo.c */
#define CL_CHECK(_expr)\
do {\
    cl_int _err = _expr;\
    if (_err == CL_SUCCESS)\
        break;\
    fprintf(stderr, "[%d@%s] OpenCL Error: '%s'\
                    returned %d(%s)!\n", \
        __LINE__, __FILE__, #_expr, (int)_err, \
        ErrCLtoStr((int)_err));\
    exit(1);\
} while (0)

#define CL_CHECK_ERR(_expr)\
({\
    cl_int _err = CL_INVALID_VALUE;\
    typeof(_expr) _ret = _expr;\
    if (_err != CL_SUCCESS) {\
        fprintf(stderr, "[%d@%s] OpenCL Error:\n '%\
                    s' code %d (%s)!\n", \
            __LINE__, __FILE__, #_expr, (int)_err, \
            ErrCodeToStr((int)_err));\
        exit(1);\
    }\
    _ret;\
})
```

Listagem 4: Macros `CL_CHECK` e `CL_CHECK_ERR`

O Programa OpenCL

Lista de plataformas e lista de dispositivos

O primeiro passo num programa OpenCL respeita a obtenção da lista de plataformas OpenCL existente na máquina. Para o efeito recorre-se à função `clGetPlatformIDs` que preenche um vetor do tipo `cl_platform_id` com os identificadores das plataformas OpenCL disponíveis na máquina local. A função recebe três parâmetros: `num_entries` que é um inteiro sem sinal a indicar o número máximo de elementos do tipo `cl_platform_id`; o vetor `platforms` cujo endereço é indicado como segundo parâmetro comporta, e ainda o endereço da variável `num_platforms` do tipo inteiro sem sinal que é preenchida com o número de elementos do tipo `clGetPlatformIDs` que são escritos pela função no vetor `platforms`. Em caso de sucesso a função retorna `CL_SUCCESS`, caso contrário devolve um código de erro.

Obtida a lista de plataformas, o passo seguinte é a recolha da lista de dispositivos de determinada plataforma OpenCL. Para o efeito recorre-se à função OpenCL `clGetDeviceIDs`. A função recebe como primeiro parâmetro o identificador do tipo `cl_platform_id` da plataforma da qual se pretende obter a lista de dispositivos. O segundo parâmetro identifica o tipo de dispositivo pretendido, que pode ser, por exemplo, `CL_DEVICE_TYPE_CPU` para CPU, ou `CL_DEVICE_TYPE_GPU` para GPU que é o caso estudado

neste artigo. Os três parâmetros seguintes funcionam de forma similar ao observado para a função `clGetPlatformIDs`, pois servem para especificar o vetor do tipo `cl_device_id` e respetivos atributos (tamanho do vetor e número de posições preenchidas no vetor pela função). A Listagem 5 apresenta o código que procede à obtenção da lista de plataformas e à recolha do identificador de dispositivo OpenCL do tipo GPU.

```
#define MAX_PLATFORMS (5)
cl_platform_id Platforms_V[MAX_PLATFORMS];
cl_uint NumPlatforms=0;
cl_uint MaxPlatforms = MAX_PLATFORMS;
cl_device_id DeviceID;
cl_uint NumDevices=0;
(...)
// Obtém lista de plataformas OpenCL
CL_CHECK(clGetPlatformIDs(MaxPlatforms,
    Platforms_V,&NumPlatforms));
if( NumPlatforms == 0 ){
    fprintf(stderr, "[Erro] Não existem "
        "plataformas OpenCL\n");
    exit(2);
}
// Procura dispositivo OpenCL do tipo GPU
CL_CHECK(clGetDeviceIDs(Platforms_V[0],
    CL_DEVICE_TYPE_GPU, 1, &DeviceID,
    &NumDevices));
```

Listagem 5: Obtenção das listas de plataformas e de dispositivos

Criação de contexto e da fila de comandos

No terceiro passo, e encontrado um (ou mais) dispositivo OpenCL apropriado, procede-se à criação de um contexto através da função OpenCL `clCreateContext`. Para tal, é necessário preencher um vetor do tipo `cl_context_properties` que é passado como primeiro parâmetro. Na sua forma mais simples, esse vetor do tipo `cl_context_properties` é preenchido da seguinte forma: o elemento de índice 0 toma o valor de `CL_CONTEXT_PLATFORM` e o elemento de índice 1 recebe o valor do identificador da plataforma pretendida, identificador que foi previamente obtido por `clGetPlatformIDs`. Finalmente, ao terceiro elemento do vetor (índice 2) atribui-se o valor 0, sinalizando deste modo o fim do vetor. O segundo e terceiro parâmetro da função `clCreateContext` especificam o(s) dispositivo(s) OpenCL cujo identificador foi obtido através de `clGetDeviceIDs`. Dado que se pode especificar mais do que um dispositivo OpenCL, o segundo parâmetro – `num_devices` – indica o número de dispositivos OpenCL pretendidos, sendo o terceiro parâmetro – `devices` – um vetor que contém `num_devices` identificadores dos dispositivos pretendidos. O quarto parâmetro é um ponteiro para uma função que pode ser especificada pelo utilizador para tratamento de erros associados ao contexto OpenCL que se pretende criar. A função atua como *callback*, podendo ser chamada de forma assíncrona pela implementação de OpenCL na ocorrência de um erro. Por razões de simplicidade e espaço, neste artigo não se recorre a este mecanismo, indicando-se `NULL` para este parâmetro e para o seguinte. De facto, o parâmetro seguinte está ainda relacionado com a função

de *callback*, tratando-se de um ponteiro, a especificar pelo programador, para uma zona de memória que contém dados a serem passados à função de *callback*. Finalmente, o último parâmetro corresponde à passagem por referência da variável para preenchimento do código de estado da execução da operação de criação de contexto. Caso a operação seja bem sucedida, a função **clCreateContext** devolve um identificador de contexto do tipo **cl_context**.

```
cl_context_properties Props_V[3];
cl_context Contexto;
(...)
// Vetor com propriedades para criação de contexto
// (último elemento deve obrigatoriamente conter 0)
Props_V[0]= CL_CONTEXT_PLATFORM;
Props_V[1]= (cl_context_properties) Pltformas_V[0];
Props_V[2]= 0;
// Cria um contexto com o dispositivo GPU
Contexto = CL_CHECK_ERR(clCreateContext(Props_V,1,
                                     &DeviceID,NULL,NULL,&_err));

// Cria fila de espera de comandos
FilaComandos = CL_CHECK_ERR(clCreateCommandQueue(
                        Contexto, DeviceID, 0, &_err));
```

Listagem 6: Criação do contexto e da fila de espera de comandos

Obtido um contexto, procede-se à criação da fila de espera de comandos, recorrendo-se à função OpenCL **clCreateCommandQueue**. A função recebe como primeiro parâmetro o identificador de contexto do tipo **cl_context**, e como segundo parâmetro um identificador de dispositivo do tipo **cl_device_id**. No terceiro parâmetro são especificadas as propriedades que se pretendem para a fila de espera. As propriedades suportadas são **CL_QUEUE_PROFILING_ENABLE** e **CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE**. A primeira é necessária para a medição do tempo despendido na execução dos comandos passados à fila de espera (*profiling*), ao passo que a segunda possibilita especificar que as operações OpenCL sejam executados por ordem diferente da que são inseridos na fila de espera. Caso se pretenda fazer uso de ambos as propriedades, os dois identificadores devem ser combinadas com o operador **|** (OR binário da linguagem C). Por fim, o último parâmetro corresponde à passagem por referência da variável para preenchimento do código de estado da execução da função **clCreateCommandQueue**. Caso a função seja bem sucedida, é retornado um identificador do tipo **cl_command_queue**.

Carregamento, compilação e criação do kernel OpenCL

O passo seguinte corresponde à criação do identificador de programa do tipo **cl_program**. Para o efeito, recorre-se à função **clCreateProgramWithSource**, através da qual se especifica o código fonte do kernel. O primeiro parâmetro da função especifica o identificador de contexto. O segundo e terceiro parâmetro estão ligados entre si: o segundo parâmetro é um inteiro que indica o número de strings apontadas pelo terceiro parâmetros. O terceiro parâmetro é uma das originalidades do

OpenCL: trata-se de um vetor de strings que contém o código fonte do kernel que se pretende usar no dispositivo OpenCL. De facto, no OpenCL o kernel é especificado através de um vetor de strings. É habitual que o kernel seja apresentado no formato string tal como é mostrado na Listagem 7. Contudo, este formato é pouco legível, dificultando a tarefa do programador, pelo que se optou neste artigo por colocar o código fonte do kernel num ficheiro à parte com o respetivo nome a ter a extensão **.cl**. O conteúdo de um ficheiro **.cl** é carregado para um vetor de strings através da função **FichParaVetorStr** (Listagem 9) que aloca e preenche um vetor de strings a partir de um ficheiro de texto (uma string por cada linha do ficheiro). A função preenche ainda uma variável inteira com o número de strings do vetor. Importa contudo notar que a separação do código fonte do *kernel* OpenCL do restante código da aplicação, apesar de aumentar a legibilidade, reduz a robustez da aplicação (a execução fica dependente da existência do ficheiro com o código do kernel) e força a que o código fonte do kernel seja fornecido em formato de texto, o que poderá não ser desejável num ambiente de produção. O quarto parâmetro da função **clCreateProgramWithSource** é (mais) um exemplo da complexidade e verbosidade que invadem a API do OpenCL: trata-se de um vetor que contém o número de caracteres de cada string do vetor que contém o código fonte do kernel. Assim, o elemento de índice 0 indica o número de caracteres da primeira string que contém o código fonte do kernel, o elemento de índice 1 contém o número de caracteres da segunda string do código fonte do kernel e assim sucessivamente. Este parâmetro só é empregue caso as strings do código fonte do kernel não sejam terminadas por **'\0'**. No caso do código do kernel ser especificado através de strings em formato C, este parâmetro é colocado a **NULL**. Finalmente, o quinto parâmetro da função corresponde ao código que reflete o sucesso/insucesso da execução. Caso a execução decorra normalmente, a função retorna um identificador do tipo **cl_program**.

```
/**
 * Código fonte de Inverte1D especificado via
 string */
const char *CodigoPrograma =
 "__kernel void Inverte1D(__global const char *
 inputImg, \n\"
 \"                __global char * outputImg, \n\"
 \"                int largura, int altura)\n\"
 \"{ \n\"
 \"    int NumPixels = largura*altura; \n\"
 \"    unsigned int i = get_global_id(0); \n\"
 \"    unsigned int TotalThreads = get_global_size
 (0); \n\"
 \"    while(i<NumPixels){ \n\"
 \"        outputImg[i] = 255-inputImg[i]; \n\"
 \"        i += TotalThreads; \n\"
 \"    } \n\"
 \"} \n\";
```

Listagem 7: Apresentação alternativa do Kernel Inverte1D recorrendo-se a uma string constante

```
char **KernelStr;
int NumLinhas;
cl_program Programa;
```


A PROGRAMAR

PROGRAMAÇÃO EM OPENCL

```
...
// Cria vetor de string KernelStr a partir do
//ficheiro
KernelStr = FichParaVetorStr(
    "kernel_inverte1D.cl",&NumLinhas);
// Cria o programa a partir do código fonte
Programa = CL_CHECK_ERR( clCreateProgramWithSource(
    Contexto, NumLinhas,
    (const char**)KernelStr, NULL, &_err) );
```

Listagem 8: Carregamento do ficheiro de kernel e criação do objeto do tipo `cl_program`

Carregado o código fonte do kernel, o passo seguinte é compilá-lo. De facto, é uma das outras originalidades do OpenCL: compilar o kernel durante a execução do processo hospedeiro. Essa abordagem permite, pelo menos teoricamente, o uso do compilador mais apropriado ao dispositivo OpenCL, dado que o compilador se encontra integrado no *driver* OpenCL do dispositivo. Acresce-se que a atualização do driver pode também comportar a atualização do compilador. A compilação do código fonte do kernel efetua-se através da função OpenCL `clBuildProgram`. Para tal, especifica-se como primeiro parâmetro o identificador de programa obtido no passo anterior (via `clCreateWithProgramWithSource`). O segundo e terceiro parâmetro atuam como um par, com o segundo parâmetro a indicar o número de elementos existentes no vetor de dispositivos OpenCL, vetor esse especificado pelo terceiro parâmetro. O segundo e o terceiro parâmetro podem ser deixados a `NULL`, levando a que o kernel seja compilado para todos os dispositivos associados ao programa. O quarto parâmetro é uma string com as opções de compilação a serem passadas ao compilador. As opções disponíveis estão definidas na norma OpenCL. Por exemplo, a opção `-cl-opt-disable` desativa todas as otimizações. Finalmente, os últimos dois parâmetros da função correspondem, respetivamente, a um ponteiro para função de *callback* e a um ponteiro para dados a serem passados a essa função de *callback*. Quando é especificado função de *callback*, a função `clBuildProgram` retorna imediatamente, sendo a função de *callback* ativada pelo OpenCL quando a compilação termina, independentemente do sucesso ou não da operação de compilação. Ambos os parâmetros associados ao mecanismo de *callback* podem ser especificados a `NULL` tendo por efeito desativar o mecanismo. Finalmente, no caso da compilação falhar, é possível aceder à saída do compilador com a indicação dos erros e eventuais avisos através da função OpenCL `clGetProgramBuildInfo` conforme mostrado na Listagem 10.

Compilado o kernel, procede-se à criação do objeto kernel através da função OpenCL `clCreateKernel`. O primeiro parâmetro da função corresponde ao identificador de programa devolvido pela função `clCreateWithProgramSource`. O segundo parâmetro é uma string que contém o nome do kernel. O terceiro parâmetro retorna o código de estado de execução da função. Caso a execução seja bem sucedida, a função devolve um identificador de kernel do tipo `cl_kernel`.

Concluída a fase que envolve a criação de contexto, a criação da fila de espera, a compilação e a criação do objeto kernel, transita-se para uma segunda fase, dependente do kernel que se pretende executar. Nesta fase, ocorre 1) a alocação e preenchimento das zonas de memória do dispositivo OpenCL sobre a qual o kernel vai atuar; 2) a execução do kernel e finalmente 3) a cópia dos resultados do espaço de endereçamento do dispositivo OpenCL para a máquina hospedeira.

```
/*
 * Nota: a entidade chamante é responsável
 * por libertar a memória alocada nesta função.
 */
char **FichParaVetorStr(char* nomeFich, int
*numElms){
#define MAX_SIZE (1024)
    char Linha_S[1024];
    char *TodasStrs, **VecStr;
    FILE *F;
    int NumLinhas, LinhaCorrente;
    F = fopen(nomeFich,"rt");
    if(F==NULL){
        fprintf(stderr,
            "Impossível abrir%s'\n",nomeFich);
        exit(3);
    }
    NumLinhas=0;
    while(!feof(F)){
        if( fgets(Linha_S,sizeof(Linha_S),F) == NULL)
        {
            break;
        }
        NumLinhas++;
    }
    /* Aloca memória para o vetor de strings */
    VecStr=(char**) malloc(sizeof(char*)
*numLinhas);
    /* Um bloco de memória para todas as strings */
    TodasStrs = (char*) malloc(
        sizeof(char)
        *MAX_SIZE*NumLinhas);

    rewind(F);
    LinhaCorrente = 0;
    while(!feof(F)){
        if(fgets(Linha_S, sizeof(Linha_S),F) == NULL)
        {
            break;
        }
        VecStr[LinhaCorrente] =
            TodasStrs+LinhaCorrente*MAX_SIZE;
        strncpy(VecStr
            [LinhaCorrente],Linha_S,MAX_SIZE);
        LinhaCorrente++;
    }
    fclose(F);
    *numElms = NumLinhas;
    return VecStr;
}
```

Listagem 9: Função `FichParaVetorStr` (leitura de ficheiro para vetor de strings)

```
cl_int Ret;
cl_kernel kernel;
...
// Compila o programa
Ret = clBuildProgram
(Programa,0,NULL,NULL,NULL,NULL);
```

```
if( Ret != CL_SUCCESS){
    char Build_S[4096];
    fprintf(stderr, "[ERRO] clBuildProgram '%s' "
        "(code: %d)\n", ErroCLtoStr(Ret), Ret );
    // solicita log da compilação
    clGetProgramBuildInfo( Programa, DeviceID,
        CL_PROGRAM_BUILD_LOG, sizeof
        (Build_S), Build_S, NULL);
    fprintf(stderr, "[ERRO] log: '%s'\n", Build_S);
    exit(4);
}
// Especifica o kernel que deve ser executado
kernel = CL_CHECK_ERR( clCreateKernel( Programa,
    "InverteID", &_err));
```

Listagem 10: Compilação e criação do kernel

Alocação de memória no dispositivo OpenCL

No exemplo da inversão de imagem, são necessários dois blocos de memória no dispositivo OpenCL: um bloco para a imagem a inverter e outro para a imagem invertida. São esses dois blocos de memória que são acedidos pelo kernel OpenCL. A alocação de memória no espaço de endereçamento de um dispositivo OpenCL faz-se através da função **clCreateBuffer**. O primeiro parâmetro identifica o contexto. O segundo parâmetro especifica o modo de alocação pretendido que pode ser, entre outros, **CL_MEM_READ_WRITE** (bloco de memória pode ser lido e escrito), **CL_MEM_WRITE_ONLY** (o dispositivo OpenCL apenas pode escrever na memória, não podendo efetuar operações de leitura) e **CL_MEM_READ_ONLY** (o dispositivo OpenCL apenas pode ler o bloco de memória). Na aplicação de inversão de imagem, a memória para a imagem original é alocada em modo **CL_MEM_READ_ONLY**, ao passo que a memória que se destina a receber a imagem invertida é alocada em modo **CL_MEM_WRITE_ONLY**. O terceiro parâmetro de **clCreateBuffer** indica o tamanho, em bytes, da memória que se pretende alocar. O quarto parâmetro corresponde a um ponteiro para uma zona de memória do processo hospedeiro, sendo que a interpretação do ponteiro está dependente do modo indicado no segundo parâmetro. Assim, se for acrescentado, através do operador **OR** binário **|**, a opção **CL_MEM_COPY_HOST_PTR**, o conteúdo da memória (do processo hospedeiro) apontada pelo ponteiro é copiado para a memória a ser alocada no dispositivo OpenCL. Se for acrescentada (novamente através do operador **OR** binário **|**) a opção **CL_MEM_USE_HOST_PTR** ao modo pretendido, então o endereço especificado no quarto parâmetro é empregue como *cache* em eventuais posteriores operações de cópia. Por razões de simplicidade, o exemplo focado neste artigo especifica este parâmetro a **NULL**. O quinto parâmetro é a já habitual variável atualizada pela função com o código de estado resultante da execução. Finalmente, a função **clCreateBuffer** retorna um identificador do tipo **cl_mem**.

A operação seguinte consiste na cópia da imagem a processar que reside num vetor da memória da máquina hospedeira para o vetor alocado no passo anterior. Para o efeito é empregue a função OpenCL **clEnqueueWriteBuffer** que recebe como primeiro parâmetro o identificador da fila de comandos OpenCL.

O segundo parâmetro corresponde ao identificador da zona de memória do dispositivo OpenCL (tipo **cl_mem**) para a qual se pretende escrever. O parâmetro seguinte é booleano, permitindo configurar a cópia como bloqueante (a função só retorna quando a cópia estiver concluída) através do valor **CL_TRUE**. Se for especificado o valor falso, isto é, **CL_FALSE**, a função tem comportamento não bloqueante. O parâmetro seguinte indica o deslocamento (*offset* na designação anglo-saxónica), em bytes, em relação ao início da zona de memória a partir da qual se deve iniciar a escrita. Por exemplo, indicando-se 16, a escrita efetuar-se-á a partir do 17º byte da memória de destino. Neste artigo não se emprega essa funcionalidade, indicando-se zero. O quinto parâmetro especifica o número de bytes a serem copiados para a zona de memória do dispositivo OpenCL. O parâmetro seguinte corresponde ao endereço da zona de memória do processo da máquina hospedeira cujo conteúdo se pretende copiar (indicado pela variável *ImageIn_h* na Listagem 11). Finalmente, os três últimos parâmetros estão associados a funcionalidades avançadas, nomeadamente à possibilidade de ser especificada uma lista de eventos que devem ocorrer antes que a função **clEnqueueWriteBuffer** possa proceder à cópia dos dados. Essa funcionalidade pode ser desativada especificando 0, **NULL** e **NULL**, respetivamente, para os três últimos parâmetros.

```
size_t TamImagem;
cl_mem input, output;
...
// Cria os vetores input (imagem a inverter) e
// output (imagem invertida) na memória do
// dispositivo
TamImagem = Largura * Altura * sizeof(unsigned
    char);
input = CL_CHECK_ERR(clCreateBuffer(Contexto,
    CL_MEM_READ_ONLY, TamImagem, NULL, &_err));
output = CL_CHECK_ERR(clCreateBuffer(Contexto,
    CL_MEM_WRITE_ONLY, TamImagem, NULL, &_err));
// Carrega imagem a inverter na memória input
CL_CHECK(clEnqueueWriteBuffer(FilaComandos, input,
    CL_TRUE, 0, TamImagem, ImageIn_h, 0, NULL, NULL));
```

Listagem 11: alocação de memória no dispositivo OpenCL e cópia da imagem original para o bloco input

Preparação e execução do Kernel

Chegado a esta fase, apenas falta um passo para que o kernel esteja pronto a ser executado: carregamento dos argumentos necessários ao kernel. Para o efeito, cada argumento é especificado através da função OpenCL **clSetKernelArg**. O primeiro parâmetro dessa função recebe o identificador do kernel (anteriormente criado através de **clCreateKernel**), o segundo parâmetro corresponde a um valor inteiro que especifica o índice posicional do argumento no kernel. Por exemplo, para se definir o terceiro argumento do kernel especifica-se 2 para o índice posicional. O terceiro parâmetro corresponde ao tamanho, em bytes, do parâmetro que se pretende passar ao kernel. Finalmente, o quarto parâmetro corresponde ao endereço de memória no qual se encontra o

A PROGRAMAR

PROGRAMAÇÃO EM OPENCL

argumento que vai ser copiado para o kernel. Importa notar que a função **clSetKernelArg** deve ser chamada tantas vezes quanto o número de argumentos do kernel.

O passo seguinte corresponde ao lançamento da execução do kernel, operação efetuada através da função OpenCL **clEnqueueNDRangeKernel**. Os dois primeiros parâmetros da função servem para especificar, respetivamente, o identificador da fila de comandos e o identificador do kernel. O terceiro parâmetro serve para indicar o número de dimensões (*dim*) da geometria de execução, aplicando-se tanto aos *work-items* como aos *work-groups*. O parâmetro seguinte é um vetor que deve ter tantos elementos quanto o número de dimensões especificados no parâmetro anterior. O conteúdo do vetor é interpretado como os deslocamentos (um por cada dimensão) empregues para calcular o identificador global de cada work-item (relembre-se que no kernel, o identificador global na dimensão **dim** obtém-se através da função **get_global_id**(int dim)). Se esse vetor for especificado a NULL, então o identificador global de cada item é calculado a partir do referencial (0,...,0). O parâmetro seguinte indica a geometria de execução global, definindo o número de work-groups para cada dimensão. Para tal, o parâmetro é um vetor em que o elemento de índice *i* especifica o número de work-groups na *i*-ésima dimensão, tendo o vetor um elemento por dimensão. O parâmetro seguinte define a geometria de execução local, novamente através um vetor de inteiros com *dim* elementos que indica o número de work-items para cada dimensão. Cada elemento de índice *i* indica, para a dimensão *i*, o número de work-items existentes em cada work-group. Caso seja especificado a NULL, o OpenCL encarrega-se de definir uma geometria de execução local com base na geometria de execução global e no próprio dispositivo OpenCL. Finalmente, os três últimos parâmetros da função **clEnqueueNDRangeKernel** disponibilizam a mesma funcionalidades dos últimos três parâmetros da função **clEnqueueWriteBuffer**.

```
size_t TamImagem;
cl_mem input, output;
...
// Cria os vetores input (imagem a inverter) e
// output (imagem invertida) na memória do dispositivo
TamImagem = Largura * Altura * sizeof(unsigned
char);
input = CL_CHECK_ERR(clCreateBuffer(Contexto,
CL_MEM_READ_ONLY, TamImagem, NULL, &_err));
output = CL_CHECK_ERR(clCreateBuffer(Contexto,
CL_MEM_WRITE_ONLY, TamImagem, NULL, &_err));
// Carrega imagem a inverter na memória input
CL_CHECK(clEnqueueWriteBuffer(FilaComandos, input,
CL_TRUE, 0, TamImagem, ImageIn_h, 0, NULL, NULL));
```

Listagem 12: Lançamento da execução do kernel

É importante notar que a execução do kernel é assíncrona. Assim, **clEnqueueNDRangeKernel** apenas acrescenta a execução do kernel à fila de comandos, devolvendo de imediato a execução ao processo hospedeiro. Na Listagem 12, o kernel *Inverte1D* é lançado para execução numa geometria unidimen-

sional com 64 work-groups, existindo 32 work-items por work-group.

Cópia dos resultados e encerramento

Terminada a execução do kernel, resta apenas copiar os resultados do espaço de endereçamento do dispositivo OpenCL para o processo hospedeiro. Dado a execução do kernel ser assíncrona, é necessário garantir que a cópia apenas ocorre após o término do kernel. Para tal, procede-se à chamada da função OpenCL **clFinish** indicando como único parâmetro o identificador da fila de comandos. A função **clFinish** bloqueia o processo hospedeiro até que todos os comandos da fila de comandos estejam terminados, atuando como um ponto de sincronização.

A cópia dos resultados produzidos pelo kernel, neste caso, a imagem invertida, efetua-se com a função OpenCL **clEnqueueReadBuffer**. A função apresenta uma tipologia de parâmetros similar à função **clEnqueueWriteBuffer**. Finalmente, procede-se à libertação dos recursos OpenCL, recorrendo a funções específicas para o efeito, tal como está documentado na segunda parte da Listagem 13.

```
// Aguarda até que o kernel tenha terminado
CL_CHECK( clFinish(FilaComandos) );

// Copia a imagem invertida da memória do dispositivo
// para a memória do programa hospedeiro
CL_CHECK( clEnqueueReadBuffer(FilaComandos, output,
CL_TRUE, 0, TamImagem, ImageOut_h, 0, NULL, NULL));

// Liberta os recursos OpenCL
CL_CHECK( clReleaseMemObject(input) );
CL_CHECK( clReleaseMemObject(output) );
CL_CHECK( clReleaseProgram(Programa) );
CL_CHECK( clReleaseKernel(kernel) );
CL_CHECK( clReleaseCommandQueue
(FilaComandos));
CL_CHECK( clReleaseContext(Contexto) );
```

Listagem 13: Libertação dos recursos OpenCL

Compilação de um programa OpenCL

Num sistema Linux no qual já se tenha instalado o OpenCL, a compilação de um programa requer que o programa seja *linkado* com a biblioteca OpenCL. Por exemplo, com o compilador *gcc*, a compilação do programa OpenCL *inverte1D.c* efetua-se através da seguinte linha de comando:

```
gcc -Wall inverte1D.c -lOpenCL -o inverte1D
```

Caso o diretório com os ficheiros **.h** do OpenCL não conste dos diretórios configurados **.h** para o compilador, torna-se necessário acrescentá-lo. No compilador *gcc*, isso consegue-se através da opção **-I** (i maiúsculo) seguido do caminho absoluto do diretório. Por exemplo, no computador empregue para este artigo, foi necessário acrescentar o caminho **/usr/local/cuda/include**, dado ser nesse diretório que se encon-

tra o diretório **CL** que por sua vez contém o ficheiro **cl.h**. A existência da palavra **cuda** no caminho indicado poderá surpreender, mas tal se deve ao facto de se estar a usar uma GPU da NVIDIA como dispositivo OpenCL: a implementação OpenCL da NVIDIA está, internamente, associada ao CUDA.

A linha de compilação completa é:

```
gcc -Wall -I/usr/local/cuda/include/   invert1D.c
-lOpenCL -o invert1D
```

Um programa OpenCL pode ainda ser compilado em sistemas Windows ou em sistemas Mac OS X. É importante lembrar que o kernel OpenCL propriamente dito (ficheiro com extensão **.cl**) apenas é compilado durante a execução do programa através do compilador disponibilizado pelo driver do dispositivo OpenCL. Deste modo, para uma correta execução da aplicação, o ficheiro **.cl** que contém o código do kernel (“**invert1D.cl**” neste caso) tem que estar no mesmo diretório do que a aplicação. Conforme referido anteriormente, numa versão de produção, o mais apropriado é que o código fonte dos *kernels* OpenCL esteja diretamente integrado no ficheiro executável (Listagem 7). A Tabela 2 resume a sequência de chamadas a funções da norma OpenCL de um típico programa OpenCL.

Função OpenCL	Ação
clGetPlatformIDs	obtém identificadores de plataformas
clGetDeviceIDs	obtém identificadores de dispositivos
clCreateContext	cria contexto
clCreateCommandQueue	cria fila de espera
clCreateProgramWithSource	cria programa
clBuildProgram	compila programa
clCreateBuffer	aloca memória no dispositivo
clEnqueueWriteBuffer	copia do processo hospedeiro para a memória do dispositivo
clSetArg	especifica argumentos do kernel
clEnqueueNDRangeKernel	executa kernel
clFinish	aguarda término kernel
clRelease...	liberta recursos

Tabela 2: Sequência de chamadas num programa OpenCL

Resultados e variações do kernel OpenCL

Quando se executa o programa **invert1D** obtém-se o negativo da imagem, mostrada na Figura 2. O resultado foi obtido empregando-se uma geometria unidimensional de 64 work-groups, cada um com 32 work-items, perfazendo pois um total

de 2048 (64x32) work-items em execução paralela. Dado que a imagem tem 262144 píxeis (512x512), cada work-item processa 128 píxeis (262144 / 2048), executando para o efeito 128 iterações do ciclo *while* (Listagem 1). Por exemplo, o work-item 0 do work-group 0 processa o píxel 0 na primeira iteração do ciclo *while*, o píxel 2048 na 2ª iteração, o píxel 4096 na 3ª iteração e assim sucessivamente. Por sua vez, o work-item 1 do work-group 0 processa o píxel 1 na 1ª iteração, o píxel 2049 na 2-a iteração, o píxel 4097 na 3ª iteração e assim sucessivamente.



Figura 2: resultado da execução de invert1D (versão invertida da imagem original)

Por forma a explicitar visualmente a organização em work-items e work-groups, procedeu-se a uma pequena alteração no kernel **invert1D.cl**, criando-se o kernel **invert1D.bordas.cl** (Listagem 14).

```
__kernel void Inverte1DBordas(
    __global const char * inputImg,
    __global char * outputImg, int largura, int
    altura)
{
    unsigned int Col = get_global_id(0);
    unsigned int Linha = get_global_id(1);
    int NumPixels = largura*altura;
    unsigned int i = get_global_id(0);
    unsigned int LocalID = get_local_id(0);
    unsigned int LocalDim = get_local_size(0);
    unsigned int TotalThreads = get_global_size(0);
    while(i < NumPixels){
        if( LocalID==(LocalDim-1) )
        {
            // Borda direita
            outputImg[i] = 0;
        }
        else{
            outputImg[i] = 255-inputImg[i];
        }
        i += TotalThreads;
    }/* while */
}
```

Listagem 14: código do kernel invert1D.bordas.cl

No kernel **invert1D.bordas.cl** os work-items que se encontram na extremidade maior do respetivo work-group colocam a zero o píxel que processam, passando esse a ter cor preta. Esses work-items são aqueles cujo identificador local (obtido através de **get_local_id(0)**) é igual ao tamanho do work-group (devolvido por **get_local_size(0)**) menos uma unidade. O resultado da execução com 64 work-groups, cada um

A PROGRAMAR

PROGRAMAÇÃO EM OPENCL

com 32 work-items é mostrado na Figura 3, sendo visíveis 16 linhas verticais (512/32=16), de cor preta, correspondente aos píxeis processados pelos work-items extremos.



Figura 3: bordos destacados numa execução com 64 work-groups e 32 work-items por work-group

Inversão de imagem com recurso a uma geometria de execução com duas dimensões

A Listagem 15 apresenta o kernel **Inverte2D** (ficheiro `inverte2D_bordas.cl`) que se destina a ser executado através de uma geometria de execução bidimensional, em que tanto os work-groups como os work-items estão organizados sobre duas dimensões (dimensão 0 e dimensão 1).

```
__kernel void Inverte2D(__global const char *
    inputImg, __global char * outputImg, int largura,
                        int altura)
{
    int Linha_0 = get_group_id(1)*get_local_size(1);
    int Linha = Linha_0 + get_local_id(1);
    int Col_0 = get_group_id(0)*get_local_size(0);
    int Col = Col_0 + get_local_id(0);
    int NumThreads_X = get_global_size(0);
    int NumThreads_Y = get_global_size(1);
    int Idx;
    while( (Linha_0 < altura) && (Col_0 < largura) )
    {
        Idx = Linha*largura + Col;
        if( (Linha < altura) && (Col < largura) ){
            if( (get_local_id(0)==0)|| (get_local_id(1)
                ==0) || ((get_group_id(0)==0)&&
                    (get_group_id(1)==0)) )
            {
                outputImg[Idx] = 0;
            }else{
                outputImg[Idx] = 255-inputImg[Idx];
            }
        }
        // Calcula o próximo bloco a processar
        Col_0 += NumThreads_X;
        Col += NumThreads_X;
        if( Col_0 >= largura ){
            Col_0 = get_group_id(0)*get_local_size(0);
            Col = Col_0 + get_local_id(0);
            Linha_0 += NumThreads_Y;
            Linha += NumThreads_Y;
        }
    }
} //while
}
```

Listagem 15: código do kernel `Inverte2D`

O kernel **Inverte2D** processa a imagem em *retângulos*, com cada *retângulo de processamento* a corresponder aos píxeis processados pelos work-items de um mesmo work-group. Assim, cada retângulo de processamento tem as dimensões do

work-group. No final de cada iteração do ciclo *while*, o retângulo de processamento é deslocado para a próxima área de intervenção. As variáveis **Linha_0** e **Col_0** representam as coordenadas da linha e da coluna, respetivamente, do primeiro work-item (com coordenadas locais (0,0)) do work-group corrente tem que processar, isto é, correspondem ao canto superior esquerdo do retângulo do work-group corrente. O cálculo do píxel a ser processado por cada work-item envolve agora duas coordenadas, representadas pelas variáveis **Linha** e **Coluna**. Deste modo, enquanto o work-group corrente estiver dentro dos limites da imagem (condição: **(Linha_0 < altura) && (Col_0 < largura)**), o work-item processa o píxel identificado por **Linha** e **Coluna**. Contudo, é ainda necessário verificar se essas coordenadas representam um píxel que está dentro dos limites da imagem (condição: **(Linha < altura) && (Coluna < largura)**), pois nas extremidades da imagem, uma parte do retângulo poderá estar dentro da imagem e outra parte fora dela, ou seja, **Col_0** e **Linha_0** (canto superior esquerdo do retângulo de processamento) poderão corresponder a píxeis da imagem, mas **Coluna** e **Linha** já estarem fora da imagem.

Terminado uma iteração, o retângulo é deslocado para a próxima posição, somando-se a **Col_0** o número de work-items na dimensão 0. Caso **Col_0** ultrapasse a largura da imagem (condição: **Col_0 > largura**), isso significa que o retângulo de processamento tem que ser movido para a posição inicial da fiada de baixo. Concretamente, o retângulo tem que descer **NumThreads_Y** linhas para baixo (**Linha_0 += NumThreads_Y**), sendo **Col_0** reiniciado com o seu valor inicial. Desta forma, o retângulo de processamento varre a imagem da esquerda para a direita e de cima para baixo.

Para explicitar a visualização dos efeitos da geometria de execução bidimensional, o kernel `inverte2D` coloca a preto os píxeis de um work-group que são processados pelos work-items com coordenada local da igual a 0 (na dimensão 0 ou na dimensão 1), bem como todos os píxeis que pertencem ao work-group que está mais à esquerda (condição: **(get_group_id(0)==0) && (get_group_id(1)==0)**).

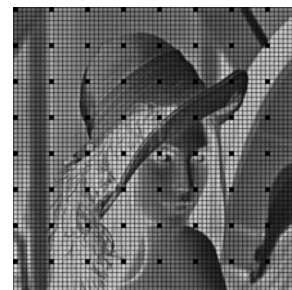


Figura 4: imagem invertida resultante da execução do kernel 2D com 64x64 work-groups e 8x8 work-items

A Figura 4 mostra a imagem invertida resultante da execução do kernel `Inverte2D` numa geometria composta por 64x64 work-groups e 8x8 work-items. Cada quadrado visível na imagem corresponde a um work-group, com dimensões

8x8 píxeis. Os quadrados totalmente a preto correspondem ao work-group (0,0), isto é ao canto superior esquerdo do retângulo de processamento. Note-se que por força da geometria de execução ser quadrada, o retângulo de processamento é ele próprio quadrado.

Notas finais

Pela necessidade de suportar dispositivos de computação paralela bastante diversificados tais como CPUs e GPUs, a norma OpenCL apresenta uma significativa complexidade que se reflete não só numa extensa API – o índice da especificação da versão 1.2 da norma lista 80 funções – mas também no elevado número de parâmetros de algumas das funções OpenCL.

Apesar da abrangência do OpenCL, a procura de desempenho requer que o programador tenha em conta as características do (s) dispositivo(s) que irá correr a aplicação. De facto, quando se procura um desempenho elevado, a execução de um kernel OpenCL num CPU com 16 cores requer uma abordagem substancialmente diferente da que deve ser seguida com uma GPU com mil ou mais cores, tendo em conta o diferente poder computacional de cada tipo de core, os diversos tipos e níveis de memória, como ainda a necessariamente diferente geometria de execução a empregar em cada caso.

“ **A norma OpenCL foi impulsionada com o intuito de uniformizar a programação para dispositivos de processamento de alto desempenho** ”

Saber mais

Apesar da extensão deste artigo, muito ficou por dizer sobre OpenCL. O artigo de Matthew Scarpino (Scarpino, 2011) abor-

da o OpenCL focando o conceito de dispositivos e a programação paralela. Foca ainda aspetos importantes como, por exemplo, os vários níveis de memória, que por falta de espaço, não foram abordados neste artigo. O guia de programação da AMD (AMD, 2012) é uma sólida referência sobre OpenCL, especialmente orientada para a programação dos dispositivos da AMD que suportam OpenCL. A referência absoluta é, obviamente, a própria norma OpenCL (Khronos Group OpenCL Working Group, 2012).

Para um conhecimento prático recomenda-se a instalação e uso de um kit de desenvolvimento (SDK), sendo que tanto a AMD, como a Intel disponibilizam SDK para a versão 1.2 da norma OpenCL que suportam os respetivos CPUs e GPUs em Windows e em Linux. A NVIDIA também disponibiliza um SDK, embora para a versão 1.1 da norma OpenCL. Útil ao programador OpenCL poderá ser a *OpenCL quick reference card* disponibilizada pelo grupo Khronos no seu sítio (Khronos OpenCL, 2011).

Bibliografia

- AMD. (Dezembro de 2012). Programming Guide / AMD Accelerated Parallel Processing OpenCL. Obtido de <http://tiny.cc/nhabrw>
- Khronos Group OpenCL Working Group. (Novembro de 2012). The OpenCL Specification/ version 1.2. Obtido de <http://tiny.cc/qjabrw>
- Khronos OpenCL. (2011). OpenCL API 1.2 Reference Card. Obtido de <http://tiny.cc/8iabrw>
- Scarpino, M. (2011). A Gentle Introduction to OpenCL. Dr Dobbs'. Obtido de <http://tiny.cc/yjabrw>
- SimpleOpenCL. (s.d.). SimpleOpenCL: library created to reduce the amount of host code needed to write an OpenCL program. Obtido em Janeiro de 2013, de <http://tiny.cc/pkabrw>
- Wikipedia. (s.d.). *Lenna*. Obtido em Janeiro de 2013, de wikipedia: <http://tiny.cc/glabrw>
- Wolf, C. (2009). Simple OpenCL demo program. Obtido em Janeiro de 2013, de <http://tiny.cc/8labrw>

«[download do código-fonte dos exemplos](#)»

AUTOR



Escrito por Patrício Domingues

Doutorado em Engenharia Informática e professor do Departamento de Eng^a Informática na Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria (IPLeiria). Leciona, entre outras, a disciplina de Computação de Alto Desempenho ao Mestrado em Engenharia Informática – Computação Móvel, onde aborda tópicos como CUDA e OpenCL. É ainda responsável pelo CUDA Teaching Center da ESTG/IPLeiria.

Accionamento de Servos Usando Arduino

Introdução:

Tal como tenho vindo a fazer nas últimas edições vou continuar a mostrar algumas das capacidades do micro-controlador Arduino.

Nesta edição apresento um artigo que aborda a movimentação e controlo de servos através do Arduino, algo muito útil para quem quer abordar mais a parte física de um projecto de robótica e electrónica.

Material necessário:

Para a implementar o exemplo que irei dar ao longo deste artigo é necessário utilizar o seguinte material.

- Arduino
- Ponte H - L293D
- Breadboard
- Jack DC
- Cablagem de Conexão

Esquema Eléctrico:

A implementação e controlo de um Servo usando Arduino já envolve um pouco mais de electrónica do que os exemplos mostrados nas edições anteriores, mas ainda assim é possível de ser realizado por pessoas sem grande experiência em electrónica e programação.

O exemplo que irei apresentar foi baseado no seguinte esquema eléctrico que eu elaborei para a conexão e controlo de dois motores servo ao Arduino de modo a conseguir obter movimento numa base do tipo “Polulu RP5 Tracked Chassis”.

O que é uma Ponte H – L293D?

Uma ponte H – L293D é um circuito integrado especialmente desenhado para facilitar a utilização de Servos DC.

Olhando para o esquema acima podemos facilmente explicar

o funcionamento de uma ponte H -L293D. Basicamente o que acontece é que quando queremos que o Servo rode para um dos lados fechamos o circuito T1 e T2, fluindo então uma corrente de 9V a 12V pelo motor até ao GND, sendo que a reversão do Servo é feita da mesma forma, mas agora fechando o circuito em T4 e T5.

A ponte H – L293D possui dois esquemas destes, um de cada lado do circuito integrado, o que permite a utilização de dois Servos em simultâneo.

Como podem ver no esquema eléctrico que desenhei o circuito eléctrico é igual para os dois servos, ou seja com a Ponte H – L293D ao centro a comandar e a receber a informação do Arduino e a distribui-la correctamente para os servos.

A energia necessária é proveniente de uma bateria de 9.2V que entra pela entrada Jack DC do Arduino, e deve ser sempre enviada pelo Jack DC devido a este ter alguns sistemas de protecção contra picos de corrente que em caso de sobrecarga poderão proteger o Arduino de problemas.

Arduino e a sua Programação:

Após a implementação do esquema eléctrico é necessário programar a parte lógica para que possa existir informação a ser enviada para a ponte H -L293D e respectivamente para os Servos.

Neste artigo irei apenas abordar uma pré-programação do Arduino de modo a que o movimento seja autónomo, sem qualquer influência do utilizador, mas no próximo artigo pegarei novamente neste exemplo e mostrarei como controlar os servos utilizando o teclado do computador como método de input.

O código que usei para a programação do Arduino foi o seguinte:

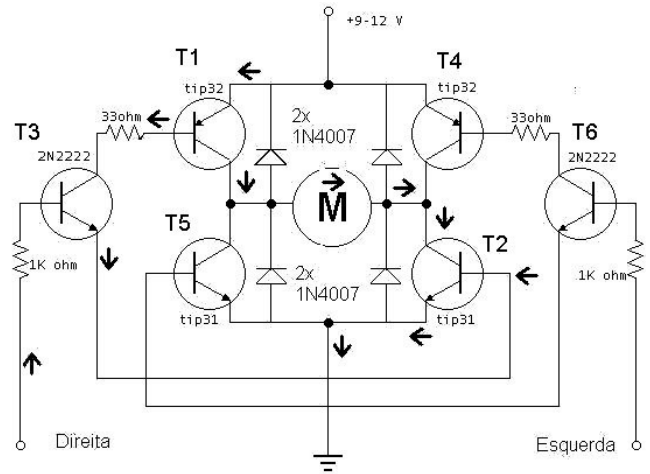
```
//Definição dos Pinos de Input do Arduino
int motorPin1 = 4;
int motorPin2 = 5;
int motorPin3 = 10;
int motorPin4 = 11;
int delayTime = 500;

void setup()
{
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
}
```

A PROGRAMAR

ACCIONAMIENTO DE SERVOS USANDO ARDUINO

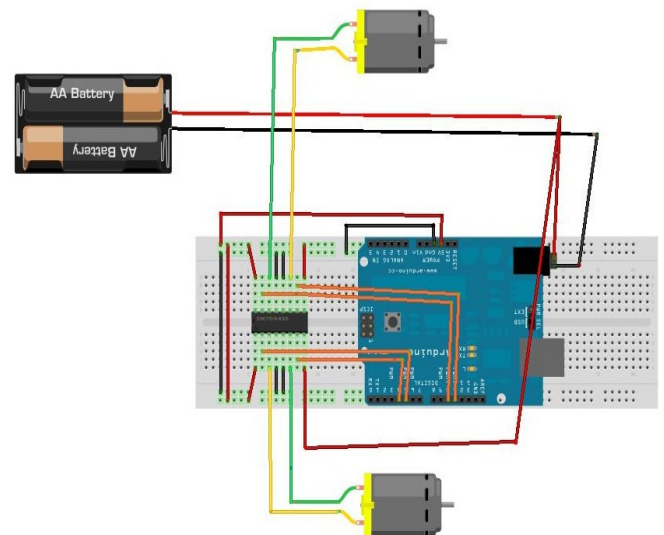
```
}  
void loop()  
{  
  //Aplicação do Código a uma sucessão de movimentos  
  frente();  
  delay(1000);  
  direita();  
  delay(1000);  
  esquerda();  
  delay(1000);  
  tras();  
  delay(1000);  
  esquerda();  
  delay(1000);  
  direita();  
  delay(1000);  
  parado();  
  delay(1500);  
}  
  
//Função de Movimento para a Frente  
void frente()  
{  
  analogWrite(motorPin1, 80);  
  analogWrite(motorPin2, 0);  
  analogWrite(motorPin3, 80);  
  analogWrite(motorPin4, 0);  
}  
  
//Função de Movimento para a Trás  
void tras()  
{  
  analogWrite(motorPin1, 0);  
  analogWrite(motorPin2, 80);  
  analogWrite(motorPin3, 0);  
  analogWrite(motorPin4, 80);  
}  
  
//Função de Movimento para a Direita  
void direita()  
{  
  analogWrite(motorPin1, 100);  
  analogWrite(motorPin2, 0);  
  analogWrite(motorPin3, 0);  
  analogWrite(motorPin4, 100);  
}  
  
//Função de Movimento para a Esquerda  
void esquerda()  
{  
  analogWrite(motorPin1, 0);  
  analogWrite(motorPin2, 100);  
  analogWrite(motorPin3, 100);  
  analogWrite(motorPin4, 0);  
}  
  
//Função Sem Movimento  
void parado()  
{  
  analogWrite(motorPin1, 0);  
  analogWrite(motorPin2, 0);  
  analogWrite(motorPin3, 0);  
  analogWrite(motorPin4, 0);  
}
```



Conclusão:

Neste pequeno tutorial expliquei de uma forma bastante simples como usar uma Ponte H –L293D para controlar Servos usando o Arduino como micro-controlador. Usando este esquema poderão criar pequenos carros comandados (numa fase seguinte irei explicar como fazer o input dos dados via teclado) ou qualquer outra coisa com movimento que o utilizador imagine.

O micro-controlador Arduino permite criar um sem número de invenções, sendo facilmente programado para que já tenha algumas noções base sobre a linguagem de programação C/C++.



AUTOR



Escrito por Nuno Santos

Curioso e autodidacta com uma grande paixão pela programação e robótica, frequenta o curso de Engenharia Informática na UTAD alimentando o sonho de ainda vir a ser um bom Engenheiro Informático. Estudante, Blogger, e moderador no fórum Lusorobótica são algumas das suas actividades. Os seus projectos podem ser encontrados em: <http://omundodaprogramacao.com/>

Um dos maiores eventos de SQL Server, o SQL Saturday, estará de volta ao nosso país no próximo dia **16 de Março no auditório da Microsoft** no Parque das Nações em **Lisboa**. Será um dia inteiro de sessões com oradores de prestígio internacional, **completamente gratuito**.

Até ao momento, o evento já conta com mais de 100 pessoas registadas e sessões submetidas por 11 MVPs, 4 Microsoft PFE's, 1 SQL Server Programa Manager e 1 evangelista de Azure, representando 12 países diferentes.



Existirão 4 tipos de sessões, DEV destinada aos desenvolvedores, BI para todos aqueles cuja paixão ou profissão é a Business Intelligence, DBA para todos os técnicos e administradores de bases de dados e Azure dedicada à plataforma de Cloud Computing homónima da Microsoft.



Realizam-se também na véspera do SQL Saturday, as Workshops de pré-conferência,

compostas por seminários e “Deep-Dives” dadas por alguns dos melhores oradores do evento. No entanto, esta oportunidade de formação avançada tem o custo de 100 euros.



Um dos primeiros workshops é apresentado por Matthew Roche, Program Manager na Microsoft, com o tema “Como tirar o melhor partido dos Serviços de Integração do SQL Server 2012”. Desde a sua introdução o SQL Server Integration Services (SSIS) fornece uma plataforma ETL de grande performance para soluções empresariais de BI e

datawarehouse. A nova versão do SQL Server 2012 traz diversas novidades e actualizações que alteram a forma como esta plataforma é usada. Neste workshop de dia inteiro, Matthew Roche vai demonstrar formas de como tirar vantagem das novas funcionalidades e ferramentas do SSIS, baseado não só na sua experiência como consultor e autor de SSIS, como também na qualidade de membro do “SSIS Product Group” da Microsoft.



Abordando “Aplicação e Migração de Bases de Dados para Windows Azure2” estará Scott Klein, SQL Azure Corporate Technical Evangelist na Microsoft. Será um seminário com um “deep-dive how-to” de como migrar aplicações e bases de dados SQL Server para o Windows Azure. Dará como exemplo

uma solução de data warehouse acompanhada por uma aplicação WEB com ASP.Net que será migrada passo a passo. Serão ainda analisados alguns cenários híbridos incluindo o uso do SQL Server em máquinas virtuais para complementar a solução PaaS e alavancar um pacote de soluções necessárias para sistemas de BI.



Finalmente, Glenn Berry, MVP e Principal Consultant na SQLSkills, conhecido especialista em HW para o SQL Server, fala sobre “Scaling SQL Server 2012. Nesta sessão são abordados os problemas típicos de escalabilidade e problemas de performance relacionados não só com o HW, mas também

com a configuração e o tipo de carga em questão. Glenn dará conselhos práticos baseados em dados concretos e na sua própria experiência.

Para obter mais informações acerca do evento está disponível o link

www.sqlsaturday.com/188/eventhome.aspx



A web em tempo real na plataforma ASP.NET

SignalR é um projecto que nasceu na equipa dos *angle brackets* e *curly braces*, ou mais especificamente ASP.NET, como um projecto *open source*. Este projecto foi recentemente incluído no conjunto de ofertas da plataforma ASP.NET, alinhado com a recente estratégia da Microsoft de unificação das tecnologias ligadas a esta plataforma numa única “marca”.

O seu objectivo começou por ser aproximar o browser e o servidor aplicacional, fornecendo uma abstracção intermédia, que nos permite abstrairmo-nos quase totalmente de como conseguimos comunicar bidireccionalmente entre browser e servidor, mas depois expandiu também o suporte para bibliotecas .NET e também aplicações Windows 8 (JavaScript).

Embora esta tecnologia ofereça mais do que uma alternativa de implementação, as principais são (para qualquer um entre JS, servidor e biblioteca .NET):

Hubs, algo simples e fácil de implementar, que torna a integração entre cliente e servidor bastante rápida, mas ainda bastante poderosa

Persistent Connection, outra opção que dá mais controlo, mas oferece menos funcionalidade pré-implementada, exige mais desenvolvimento

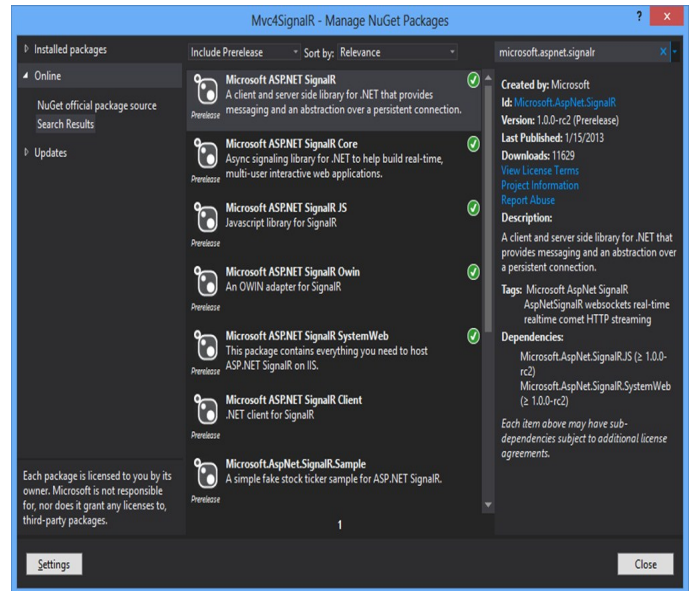
Neste artigo vou-me focar nos *Hubs* que são o modo ideal de começar a utilizar esta biblioteca.

O caso de estudo que vou implementar para demonstrar as suas funcionalidades é uma aplicação de chat em tempo real. Esta aplicação vai utilizar como *backend* uma aplicação em MVC3 (também poderia ser webforms) e *frontend* em JavaScript/HTML.

O modo mais fácil de obter as bibliotecas necessárias para utilizar o SignalR é instalar o pacote NuGet, de nome Microsoft.AspNet.SignalR, que vai criar as referências às dependências necessárias assim como incluir o(s) ficheiro(s) de JavaScript necessário(s). Para instalar, basta abrir a Package Manager Console no Visual Studio (assumindo que a extensão NuGet está instalada, caso não esteja é necessário fazê-lo) e executar o comando:

```
Package Manager Console
Package source: NuGet official package source
PM> Install-Package Microsoft.AspNet.SignalR -pre
```

Ou através da interface gráfica: (atenção na dropdown do topo seleccionar “Include Prerelease”)



No caso específico de uma aplicação ASP.NET MVC, é necessário invocar um método no application start, de modo a que um url utilizado pelo SignalR para geração de JavaScript dinamicamente fique disponível:

```
RouteTable.Routes.MapHubs();
```

Para começar, vamos definir um *Hub*, que é uma classe abstracta, da qual temos de herdar para definir um *Hub*:

```
public class ChatHub : Hub
{
}
```

Podemos também dar um nome ao nosso Hub através de um atributo, que não seja o mesmo nome da própria classe (este nome vai ser usado pelo cliente JavaScript)

```
[HubName("chat")]
public class ChatHub : Hub
{
}
```

Temos um Hub vazio para já, agora vamos passar à criação de uma página HTML que vai ser o “cliente” deste Hub:

```
<!DOCTYPE html>
<html>

  <head>

    <script src="/Scripts/jquery-1.6.4.min.js"></script>

    <script src="/Scripts/jquery.signalR-1.0.0-rc2.min.js"></script>
```


A PROGRAMAR

A WEB EM TEMPO REAL NA PLATAFORMA ASP.NET

```
<script src="/signalr/hubs" type="text/
javascript"></script>

</head>

<body>

</body>
```

A versão do jQuery pode ser actualizada para a mais recente disponível no momento, o importante de notar é que temos refenciado o ficheiro jquery.signalR-1.0.0-rc2.min.js e o caminho virtual (na realidade não existe, é gerado dinamicamente pelo SignalR em runtime) /signalr/hubs. Este código JavaScript gerado dinamicamente inclui o que é necessário para criar a ponte entre o JavaScript e o servidor sem que o programador se tenha de preocupar com isso; mais à frente vamos ver o que realmente vai ser gerado aqui.

Para acedermos ao objecto do Hub, do lado do cliente, é tão simples como:

```
var hub = $.connection.chat;
```

Em que o **chat** é o nome dado ao Hub através do atributo HubName que vimos anteriormente.

Até agora, tudo muito banal, a magia começa realmente neste objecto hub que acabámos de guardar numa variável.

Voltando à nossa classe ChatHub, vamos adicionar-lhes alguns métodos, métodos estes que vão ser automaticamente expostos através do objecto \$.connection.chat .

```
[HubName("Chat")]
public class ChatHub : Hub
{
    public void SendText(string message)
    {
        string nickName = Clients.Caller.nickName;
        Clients.Others.sendMessage(nickName, message);
    }
}
```

Por partes: os métodos públicos que são adicionados à classe ChatHub, passam a estar acessíveis no cliente através do nosso objecto \$.connection.chat. Neste caso, passamos a ter acesso à função sendText através da variável hub: hub.server.sendText("algum texto");.

A propriedade Clients permite-nos aceder a vários modos de invocar o nosso código do lado do cliente, neste caso estamos a utilizar o Others, que invoca todos os clientes, excepto o cliente que originou esta chamada ao servidor, o que faz sentido, visto que estamos a enviar uma mensagem para todos os clientes conectados, menos nós próprios. Dentro da mesma propriedade Clients, temos o objecto Caller, que se refere ao cliente que invocou o servidor. Através do Caller conseguimos aceder ao estado do cliente (neste caso estamos a aceder ao valor do campo nickName, que terá valor se este foi posto do

lado do cliente, como vamos ver a seguir), assim como invocar funções, tal como o fazemos através do Clients.Others.sendMessage.

```
var hub = $.connection.chat;

$(function ()
{
    $.connection.hub.start(function ()
    {
        hub.state.nickName = "pato";
        hub.sendMessage = function (nickName,
            message)
        {
            $("#messages").append(nickName + " >
                " + message);
        };
    });
});
```

É importante só acedermos ao hub depois de este ter iniciado, pois só aí o temos disponível. Após o document ready, invocamos o start do hub que desencadeia a ligação com o servidor, quando o callback do start é invocado significa que já conseguimos efectivamente comunicar com o servidor através do hub.

Com este JavaScript estamos a inserir a propriedade nickName com um valor (ou seja, a colocar estado no hub, que vai ser partilhado com o servidor) e a expor a função sendMessage para ser chamada pelo servidor, através do código C# que vimos anteriormente.

Dentro da função sendMessage estamos simplesmente a adicionar o texto recebido a uma div, juntamente com o nick do emissor, para identificação.

Esta acção vai ser desencadeada ao invocarmos o método SendText, criado na classe ChatHub. Para o fazermos, temos apenas de invocar a função exposta no hub do lado do cliente (criada dinamicamente através do url dinâmico referido ao início /signalr/hubs):

```
hub.server.sendText("algum texto inserido pelo
    utilizador");
```

Tipicamente, o texto virá de um input[type=text] definido no HTML, quando o utilizador clicar num botão "enviar" ou pressionar a tecla enter.

A simplicidade com que conseguimos invocar os clientes a partir do servidor, e vice-versa, é algo nunca conseguido antes.

O que a equipa que construiu o SignalR fez foi basicamente identificar um conjunto de "transportes" de comunicação entre cliente e servidor, tais como web sockets, server sent

A PROGRAMAR

A WEB EM TEMPO REAL NA PLATAFORMA ASP.NET

events, forever frame e long polling. O que acontece é que assim que uma ligação é iniciada a partir do cliente, existe um pedido inicial ao servidor, chamado de handshake, que identifica o cliente ao servidor e a seguir é efectuado outro pedido que define qual o transporte utilizado daí em diante, que irá ser o melhor possível que o cliente suporte, embora também seja possível forçar um transporte específico.

A imagem seguinte mostra o pedido inicial de handshake, assim como o pedido seguinte que escolhe o transporte:

#	Result	Protocol	Host	URL
7	200	HTTP	localhost:60211	/signalr/negotiate?_=1358783232723
9	-	HTTP	localhost:60211	/signalr/connect?transport=serverSentEvents&connectionId...

O primeiro pedido envia um identificador único para que a resposta não venha da cache, sem qualquer corpo de mensagem, a resposta identifica o id desta ligação, assim como informação sobre que protocolo deve ser seleccionado, neste caso o servidor indica que web sockets é algo que não está disponível:

```
{"Url": "/signalr", "ConnectionId": "f93ab409-641e-4067-966a-573628426094", "KeepAlive": 20.0, "DisconnectTimeout": 40.0, "TryWebSockets": false, "WebSocketServerUrl": null, "ProtocolVersion": "1.1"}
```

A próxima opção da lista de transportes (mostrada anteriormente) são os server sent events, como é verificado no pedido a seguir ao negotiate, que vai estabelecer uma ligação com o servidor, enviado o identificador recebido, e qual o transporte seleccionado.

Este teste foi efectuado com um browser moderno, Google Chrome; se efectuarmos o mesmo teste com o Internet Explorer 8 por exemplo, o resultado será um pouco diferente:

#	Result	Protocol	Host	URL
25	200	HTTP	localhost:60211	/signalr/negotiate?_=1358784010558
26	200	HTTP	localhost:60211	/signalr/connect?transport=foreverFrame&connectionId=20...
27	200	HTTP	localhost:60211	/signalr/ping?_=1358784013936
28	200	HTTP	localhost:60211	/signalr/connect?transport=longPolling&connectionId=20353...

Neste caso, a resposta ao pedido negotiate é similar, mas de

seguida, como a biblioteca JavaScript do SignalR detecta que o browser não suportará os server sent events, o connect é feito com o transporte foreverFrame, e depois este também não é suportado ou não é conseguido por algum motivo e é degradado para um long polling.

Voltando à implementação dos Hubs, após este aparte em detalhe que mostra como o SignalR funciona internamente em alguns pontos, podemos analisar os vários métodos que existem para comunicar com o cliente a partir do servidor, além do método que já vimos, Clients. Others, existem vários outros métodos na propriedade Clients, tais como:

- All: ao invocarmos utilizando a propriedade All, vamos invocar uma função em todos os clientes
- AllExcept(string connectionId): invocar todos os clientes excepto o que tenha o id fornecido como parâmetro
- Caller: invocar o cliente que originou esta chamada ao servidor
- Client(string connectionId): invocar exclusivamente o cliente cujo id seja o fornecido como parâmetro

Além destas funcionalidades e das já expostas anteriormente, existe uma funcionalidade que permite agrupar conexões em grupos, imaginando neste case study da aplicação de chat, um caso de uso seriam salas de chat. Importante é ter em conta que o programador é responsável por manter uma lista de grupos e quem lhes pertence, visto que o SignalR não fornece esta funcionalidade, simplesmente disponibiliza os métodos Add(connectionId, groupName) e Remove(connectionId, groupName) que servem para adicionar e remover ligações de um determinado grupo respectivamente.

Após ler este artigo, a melhor maneira de prosseguir será ir ao GitHub, verificar as samples da própria equipa do SignalR, ler a documentação em detalhe e iniciar um projecto real com esta tecnologia, que pode reinventar muito do que já existe, estreitando o espaço que existe entre o cliente e servidor.

GitHub: <https://github.com/SignalR/SignalR>

AUTOR



Escrito por **Ricardo Rodrigues**

Técnico Nível III em Informática/Gestão pela Fundação Escola Profissional de Setúbal, tendo ingressado após na FCT da Universidade Nova de Lisboa. Posteriormente frequentou vários cursos da Microsoft em diversas áreas como Windows Forms, ASP.NET, Securing .NET Applications, WCF, WWF, Web Services e COM+ tendo obtido as certificações MCP .NET 2.0, MCAD .NET 1.1, MCSA .NET 1.1, MCPD Windows, Web e Distributed Applications e MCPD - Enterprise Applications Developer. ([MCP Profile](#)) Contribui activamente em comunidades como [StackOverflow](#) e também possui um blog/twitter com temática relacionada: [Blog](#) / [@ricmrodrigues](#)

Bubble Sort – A Técnica da Bolha

Depois de cimentadas as bases, organizar arrays torna-se o principal desafio em qualquer primeira linguagem.

MOMENTO I

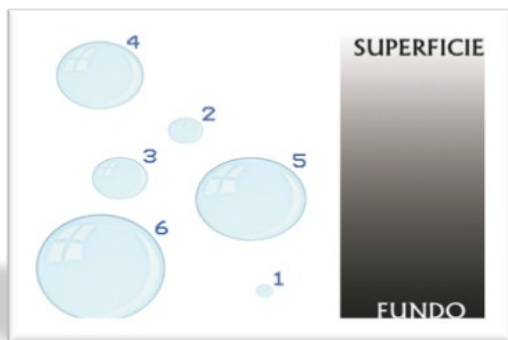
É a reciclar código, vezes sem conta, que aspirantes a programadores encarçam a lógica. E cedo percebem que o produto de um problema complexo não passa por uma só ferramenta, mas pela simbiose de múltiplas. Essa circunstância expõe como a mecânica de um algoritmo elaborado resulta tão-somente de métodos.

Entre as mais comuns está a técnica que se inspirou no fenómeno natural da física.

Na verdade, o **Método da Bolha** (*Bubble Sort*) não se exclusiva aos estudantes, atendendo a que o podemos testemunhar, também, nos códigos de experientes IT Developers.

O método baseia-se fundamentalmente num conceito físico: a bolha de ar motiva-se à superfície pelo fato de ser composta, integralmente, por matéria menos densa que a água, fazendo com que seja “empurrada” para cima.

Este conceito pode manifestar-se numa metáfora para que entendamos as etapas que o vertebram; senão vejamos a seguinte imagem:



- No mar, percorrem desde o fundo até à superfície seis bolhas de diferentes tamanhos;
- a quantidade de ar que detêm é o fator responsável por torná-las propícias à subida;
- **por isso, as maiores chegarão primeiro.**

Considerando os pontos, vamos recriar um algoritmo que consiga veicular este comportamento físico a uma linguagem de programação. O objetivo é que funcione no contexto da organização de *arrays*.

MOMENTO II

Reconheçamos quatro unidades-tipo:

- ⇒ Bolhas ou elementos
- ⇒ Respetivas posições ou índices
- ⇒ Grupo ou Conjunto em que se inserem
- ⇒ Quantidade de oxigénio nelas detido.

Devemos testar, elemento a elemento, desde o último, se esse é maior que o seguinte.

Sendo **verdadeiro**, vai substituí-lo, deixando o inferior para trás – procedimento designado por *SWAP* – e caso contrário, se **falso**, manter-se-á posicionado no mesmo lugar, não ultrapassando nenhum valor.

Com o conjunto de bolhas anarquicamente distribuídas precisamos ordená-las acrescentemente, bebendo do que a matemática e a física nos ensinam.

NOTA

Porque este artigo se direcciona, principalmente, a jovens programadores – que à partida não conhecem o método – esta primeira vista sobre o algoritmo sofrerá ligeiros desvios, dado o propósito de um primeiro avalo, mais sensível, a garantir melhor entendimento aquando de o aprofundarmos em código. Aqueles que, contudo, já conhecem a lógica em que consiste este método poderão avançar até ao MOMENTO III.

Farei três verificações a este meio de ordenação, seguindo-o nas suas consecutivas fases. É oportuno anotar, uma vez mais, que este ainda não é o comportamento lógico do Bubble Sort, apenas um prenúncio para o que mais à frente aprofundaremos.

Ora, para uma representação mais evidente da metamorfose dos dados ao longo do progresso, teremos como referência a seguinte tabela:

CONJUNTO					
Lugar	ID_Bolha	Valor	¹ O. Inicial	¹ O. Atual	² POS
0	A	1	1	1	Null
1	B	6	2	2	Null
2	C	5	3	3	Null
3	D	3	4	4	Null
4	E	2	5	5	Null
5	F	4	6	6	Null

¹ O. – *Orientação/posicionamento/local*

² (*coordenadas da movimentação nos índices*)

A PROGRAMAR

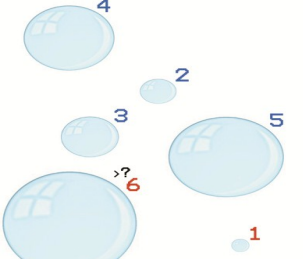
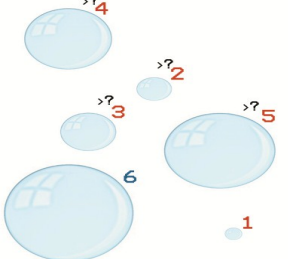
BUBBLE SORT – A TÉCNICA DA BOLHA

A tom de desafio, o leitor poderá, em papel, configurar uma nova tabela que responda objetivamente às próximas movimentações da POS. O critério de movimentação é se o valor do índice mantém posição (**null**), aumenta (+), diminui (-), e quantas vezes o fez.

Com comando **%VERIFICAR%** que encontrará adiante, o leitor deverá então comparar que dados finais apresenta a sua tabela.

Primeira verificação.

Começando pelo primeiro elemento (índice 0), devemos acrescentar-lo caso se verifique maior. E só poderemos conhecer essa superioridade comparando o seu respectivo valor com o de cada outro no conjunto.

 <p>Podemos determinar se o valor no primeiro índice (1) é superior ao do segundo (6).</p> <p>Não, logo, retorna FALSO.</p> <p>Não haverá SWAP.</p>	 <p>Observamos que o valor presente na primeira bolha (1) é inferior ao das seguintes.</p> <p>E todos retornarão FALSO.</p> <p>Em nenhum caso haverá SWAP.</p>
---	--

Irá o primeiro valor subir de posição?

Não: como o primeiro valor não é superior a nenhum dos seguintes, manter-se-á na primeira posição.

Este teste vive de uma mecânica simples: a **condição**.

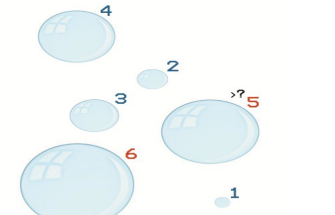
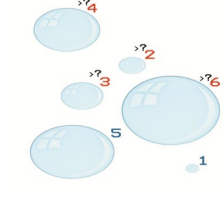
A consequência positiva ou negativa depende da quantidade de oxigênio na bolha. A influência do veredito estará, sobretudo, no futuro progressivo ou regressivo da bolha.

Com uma estrutura de decisão simples e desapropiada podemos facilmente alcançar esse veredito.

Segunda verificação.

Sabendo que a **bolha 1** (índice 0) é a inferior da conjuntura, vamos testar a próxima (índice 1). Para tal, usamos o mesmo critério de avaliação: se a **bolha 2** for maior que a seguinte, trocarão de posição.

Irá o segundo valor subir de posição?



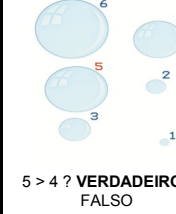


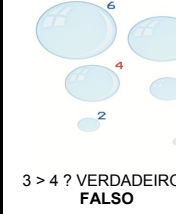



 <p>O segundo índice pergunta se o seu valor (6) é, ou não, maior que o próximo (5).</p> <p>Sim, logo, retorna VERDADEIRO.</p> <p>Haverá SWAP.</p>	 <p>O valor trocou posição, ficando o superior no índice de cima, trazendo para baixo o inferior.</p> <p>Todos retornarão VERDADEIROS.</p> <p>Em todos os casos haverá SWAP.</p>
--	---

Sim: ao contrário da primeira análise, o valor será superior a todos os seguintes, logo, subirá para a última posição.

Terceira verificação.

Estes testes consecutivar-se-ão até ter percorrido todos os índices do conjunto, sempre atraindo o maior acima e deixando para trás o valor mais pequeno.

Deste binário de exemplos podemos extrair vereditos devolvidos e não devolvidos. Sempre que o teste devolva **verdadeiro**, o computador **tem permissão** para fazer trocar os elementos em questão.

 <p>5 > 3 ? VERDADEIRO : FALSO</p>	 <p>5 > 2 ? VERDADEIRO : FALSO</p>	 <p>5 > 4 ? VERDADEIRO : FALSO</p>
 <p>5 > 6 ? VERDADEIRO : FALSO</p>	 <p>3 > 2 ? VERDADEIRO : FALSO</p>	 <p>3 > 4 ? VERDADEIRO : FALSO</p>
 <p>2 > 3 ? VERDADEIRO : FALSO</p>	 <p>4 > 5 ? VERDADEIRO : FALSO</p>	 <p>PRODUTO</p>

Como previsto, evidenciamos que os valores se ordenaram.

A PROGRAMAR

BUBBLE SORT – A TÉCNICA DA BOLHA

Teste

Se embrionarmos com a tabela anterior estes resultados, comparando-os, concluímos que movimentos se fizeram ao longo dos testes.

CONJUNTO					
Lugar	ID_Bolha	Valor	*O. Inicial	*O. Atual	*POS
0	A	1	1	1	Null
1	E	2	6	2	-3
2	D	3	5	3	-1
3	F	4	3	4	-2
4	C	5	2	5	+2
5	B	6	4	6	+4

É possível evidenciar as dissemelhanças entre a orientação inicial e a que obtivemos com o terminar do processo. À mesma conclusão chegamos se prestarmos foco à mixórdia que se concluiu na coluna **ID_Bolha**, ordenada alfabeticamente na anterior tabela.

Podemos ainda concluir os incrementos e decrementos que ocorreram na coluna de posição movível.

%VERIFICAR%

A troca de elementos

Para este resultado, existiu, nos devidos casos, um ingrediente que assumiu as trocas de elemento sempre que o teste retornou **verdadeiro**.

SWAP é a troca de elementos transversal às linguagens. Existe a necessidade de conhecê-la enquanto progredimos no terreno dos vetores e das condições, ora devido à frequência nas técnicas que os envolvem, ora pelo seu recurso em exames universitários.

Exercício lógico:

Duas mãos são constantes no nosso problema. Cada uma segura valores diferentes: mão esquerda - pedra branca; mão direita - pedra preta.

Em objetivo está levar as pedras às respetivas mãos contrárias com as seguintes limitações:

- ⇒ As pedras não se tocam.
- ⇒ Não se pousam em nada que não seja mão.
- ⇒ Não podemos portar duas pedras numa mão só.
- ⇒ É impossível projetar ao ar ou deixar cair alguma.



Resolução:

Chama-se uma terceira mão que segura temporariamente a pedra preta, enquanto a mão que a guardava (direita) recebe agora a pedra branca. A mão auxiliar dá então a pedra preta à mão que anteriormente segurava a pedra branca (esquerda).

Terceira mão vem auxiliar as duas, servindo de suporte



temporário

O conceito é simples: uma troca por troca de valores em duas unidades de suporte. Contudo, para criar a oportunidade de passagem usamos uma terceira unidade, auxiliar, responsável por guardar um dos valores durante o ato do câmbio.

Vamos praticá-lo num algoritmo.

MOMENTO III

Em programação, a lógica para organizar alfabética ou numericamente, crescente ou decrescentemente, não é distante daquilo que já vimos.

Para que seja possível comparar no array um índice e o seu sucessor, precisamos abrir o estojo e pousar todas as ferramentas sobre a mesa. Este é o momento de pôr à prova tanto o conhecer das bases lógicas adquiridas como o estudo incansável das condições e ciclos.

Sendo a linguagem donde provieram PHP, Python e Java, utilizarei ANSI C na recriação da técnica da bolha.

BUBBLE SORT (prática)

Ferramenta	Função
Array ou Vetor	Armazena valores ao longo dos seus índices
Contador Principal	Percorre o array e expõe o primeiro índice
Contador Auxiliar	Permite o teste expondo o segundo índice
Var Temporária	Guarda temporariamente um valor (SWAP)
Constante	Será o tamanho do array e o alcance do ciclo

A PROGRAMAR

BUBBLE SORT – A TÉCNICA DA BOLHA

Ao programar, perante um “invisível” erro lógico que persistia há horas, vi-me querer testar à lupa a mecânica do algoritmo em causa; para isso, coloquei-o formato tabela e decifrei-o. Rapidamente capturei a fonte do problema. Com sucesso, tenho vindo a radarizar todos os erros lógicos através desta metodologia certo dia improvisada.

Das mais importantes ferramentas Microsoft, Excel, se vocacionado, consegue assumir visualmente a perspetiva esquelética de um programa. É um método de algoritmia pouco comum (ou nulo) entre programadores. Todavia é, por experiência, eficaz na maioria dos algoritmos. Trata de ser um teste cirúrgico ao código, onde analisará tudo o que instruímos ao computador.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															

BUBBLE SORT - A TÉCNICA DA BOLHA									
CONTADORES	ÍNDICES DO ARRAY " val[WIDE] "								
DECLARADOS	0	1	2	3	4	5	6	7	8
int count	8	5	9	12	0	-5	-3	5	56
int c_aux	8	5	9	12	0	-5	-3	5	56

# define WIDE 9	temp =	0
int val[WIDE]	count =	0
int temp = 0	c_aux =	0

int count	
int c_aux	

Para que num código extenso não tenhamos que alterar, um a um, todos os valores à mínima redimensão do array, não lhe atribuiremos diretamente quantidade numérica. <! # define WIDE 9 !> [B13] será a invocação do macro que vai definir quantos elementos possui o vetor <! Val[WIDE] !> [B14], e, conseqüentemente, o máximo de voltas que os ciclos efetuam.

Dois contadores (<! int count, int c_aux !>) [B17][B18] focar-se-ão em expor à função if() os índices em causa para que esta oficialize o veredito do teste. Durante o código, ambos percorrerão em simultâneo o vetor, distinguindo-se apenas com o adiantamento (+1) do c_aux.

A variável temporária <! int temp !> [B15] terá como única razão de existência surgir enquanto “terceira mão” sempre que o SWAP for chamado. Isto é, cada vez que o if() considerar o teste verdadeiro.

PSEUDOCÓDIGO

```
INICIO
    DEFINE WIDE 8;
VAR
    ARRAY INTEIRO val[WIDE];
    INTEIRO count <= 0, c_aux;
```

Linguagem C

```
# define WIDE 8
funcao_bsort (void)
{
    int val[WIDE];
    int count, c_aux;
}
```

Se a declaração de variáveis é fácil, pouco mais difícil é atribuir valores ao array; seja por input, seja por definição. Vamos, neste caso, interagir com o utilizador e pedir-lhe que introduza WIDE números.

PSEUDOCÓDIGO

```
ENQUANTO count <WIDE
    FAÇA
        MOSTRA “Digite valor:”
        val[count] <= LER
        count +1
    FIM ENQUANTO
    OU
    PARA count (0)
        DE count ATE WIDE PASSO 1
        FAÇA
            MOSTRA “Digite valor:”
            Val[count] <= LER
        SEGUINTE
```

LINGUAGEM C

```
while(count <WIDE)
{
    printf (“\nDigite valor: ”);
    scanf (“%d”, &val[count]);
    count++;
}

||
for(count=0; count<WIDE; count++)
{
    printf (“\nDigite valor: ”);
    scanf (“%d”, &val[count]);
}
```

Enquanto recebemos valores pela função scanf(), preenchemos o array val[WIDE] que contém, dentro, count. Assim, poderemos acessar através dele a todos os índices do vetor.

Após digitados os valores, assumimos hipoteticamente estes índices:

Val[WIDE] = {8, 5, 9, 12, 0, -5, -3, -5, 56}

Eis que passaremos, enfim, à ordenação! Dedicaremos um foco especial à magia do Bubble Sort, com vista posta no que já foi lecionado:

COMPARAR O NÚMERO DE UM ÍNDICE COM O NÚMERO DO ÍNDICE SEGUINTE

Um ciclo <! for(count=0; count<WIDE; count++) !>,

A PROGRAMAR

BUBBLE SORT – A TÉCNICA DA BOLHA

cujo contador é iniciado a zero, dando até **WIDE** voltas ao array **val[WIDE]**. Corre até ao final do vetor expondo ao **if()** o índice que em cada volta presencia.

Dentro deste, apenas um passo à frente, estará o segundo ciclo **<! for(c_aux=count+1; c_aux<WIDE; c_aux++) !>** que se inicia como o **count**, mas um índice acima. Assim, este segundo contador cooperará com o primeiro em nome de expor ao **if()** o segundo índice a testar.

SE O VEREDITO FOR POSITIVO SURGE UM TERCEIRO ELEMENTO PARA O SWAP

Quando a estrutura **<! if(val[count] > val[c_aux]) !>** recebe os índices apontados pelos contadores, como sabemos, analisa se o valor do primeiro é superior ao do segundo. Este teste decorrerá **WIDE** vezes.

FALSO - ignora o bloco de consequência e continua.

VERDADEIRO – declara-se neste bloco de consequência a variável **temp**:

int temp = val[count]; // É-lhe automaticamente atribuído o valor ainda presente no índice a que **count** corresponde – o maior.

val[count] = val[c_aux]; // Agora que esse valor está armazenado, o índice seguinte vai cuspir para o anterior o seu valor atual – o menor.

val[c_aux] = temp; // Por fim, a variável devolve agora ao índice acima o que tinha guardado no início – o maior.

PSEUDOCÓDIGO

```
PARA count (0)
  DE count ATE WIDE PASSO 1
    FAÇA
      PARA c_aux
        DE count+1 ATE WIDE PASSO 1
          SE val[count] > val[c_aux]
            temp <= val[count]
            val[count] <= val[c_aux]
            val[c_aux] <= temp
          FIMSE
        SEGUINTE
      SEGUINTE
```

LINGUAGEM C

```
for(count=0; count<=WIDE; count++)
{
  for(c_aux=count+1; c_aux<WIDE; count++)
  {
    if(val[count] > val[c_aux])
    {
      int temp=val[count];
      val[count]=val[c_aux];
      val[c_aux]=temp;
    }
  }
}
```

Utilizando o método algorítmico Excel, vamos fazer um preview ao que sucederá internamente no nosso programa:

(cada índice está representado pelo respetivo contador)

BUBBLE SORT - A TÉCNICA DA BOLHA									
CONTADORES	ÍNDICES DO ARRAY " val[WIDE] "								
DECLARADOS	0	1	2	3	4	5	6	7	8
int count	8	5	9	12	0	-5	-3	5	56
int c_aux	8	5	9	12	0	-5	-3	5	56

temp =	8
count =	8
c_aux =	5

temp =	8
count =	5
c_aux =	8

count (8) passa para temp; c_aux (5) para count; temp (8) para c_aux

BUBBLE SORT - A TÉCNICA DA BOLHA									
CONTADORES	ÍNDICES DO ARRAY " val[WIDE] "								
DECLARADOS	0	1	2	3	4	5	6	7	8
int count	5	8	9	12	0	-5	-3	5	56
int c_aux	5	8	9	12	0	-5	-3	5	56

temp =	0
count =	8
c_aux =	9

Este é um exemplo em como o teste devolve **FALSO**, pois **val[count]** não supera **val[c_aux]**. Por essa razão não é executado nem invocada a variável temporária no bloco consequência do **if()**.

BUBBLE SORT - A TÉCNICA DA BOLHA									
CONTADORES	ÍNDICES DO ARRAY " val[WIDE] "								
DECLARADOS	0	1	2	3	4	5	6	7	8
int count	5	8	9	12	0	-5	-3	5	56
int c_aux	5	8	9	12	0	-5	-3	5	56

temp =	0
count =	9
c_aux =	12

Ainda não se verifica o índice anterior ser maior que o adiante. Também não existe, aqui, condição para o retorno de **VERDADEIRO**. Os contadores passarão aos próximos índices à procura de expôr esses valores.

A PROGRAMAR

BUBBLE SORT – A TÉCNICA DA BOLHA

BUBBLE SORT - A TÉCNICA DA BOLHA									
CONTADORES DECLARADOS	ÍNDICES DO ARRAY " val[WIDE] "								
	0	1	2	3	4	5	6	7	8
int count	5	8	9	12	0	-5	-3	5	56
int c_aux	5	8	9	12	0	-5	-3	5	56

temp =	12
count =	12
c_aux =	0

temp =	12
count =	0
c_aux =	12

count (12) passa para temp; c_aux (0) para count; temp (12) para c_aux

Neste juízo progridem os contadores até aos restantes índices enquanto não forem atingidos **WIDE** ciclos naquele particular array.

Apresentar os valores já distinguidos é tão fácil quanto lê-los, embora **só o fazamos fora de qualquer ciclo anterior** para evitar resultados indesejados.

PSEUDOCÓDIGO

```
ENQUANTO count < WIDE
  FAÇA
    MOSTRA val[count]
    count +1
FIM ENQUANTO
OU
PARA count (0)
  DE count ATE WIDE PASSO 1
  FAÇA
    MOSTRA val[count]
  SEGUINTE
```

LINGUAGEM C

```
while(count < WIDE)
{
    printf (“\n%d”, val[count]);
    count++;
}

||
for(count=0; count<WIDE; count++)
{
    printf (“\n%d”, val[count]);
}
```

```
**REVISTA PROGRAMAR**
...:BUBBLE SORT - A Técnica da Bolha:...
[ -5 ] [ -3 ] [ 0 ] [ 5 ] [ 5 ] [ 8 ] [ 9 ] [ 12 ] [ 56 ]
```

E temos, por fim, o produto do esforço destas páginas

Modelo alfabético: cada caractere corresponde a um ID na tabela ASCII (**A** – Dec: 65, **B** – Dec: 66, ...); grande parte dos IDEs (Integrated Development Environment), estimulados pelo núcleo da linguagem, associam essa letra ao correspondente da tabela. É assim que conseguimos utilizar o método para ordenar de forma alfabética um conjunto de letras ou de nomes.

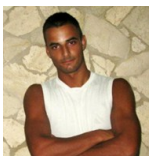
“ **o Método da Bolha (Bubble Sort) não se exclusiva aos estudantes, atendendo a que o podemos testemunhar, também, nos códigos de experientes IT Developers** ”

Logicamente, ordenações complexas levam este método a alentar-se na execução. O *Bubble Sort* original não está sobretudo desenhado para organizações alfanuméricas de elevado nível, embora possa evoluir para.

Este é o modelo genérico. Existem versões adaptadas para atingir um aumento da eficácia ou da rapidez no processo de ordenação. Modificar uma técnica para proveito de um algoritmo específico é frequente e tido como norma entre inicanos na programação.

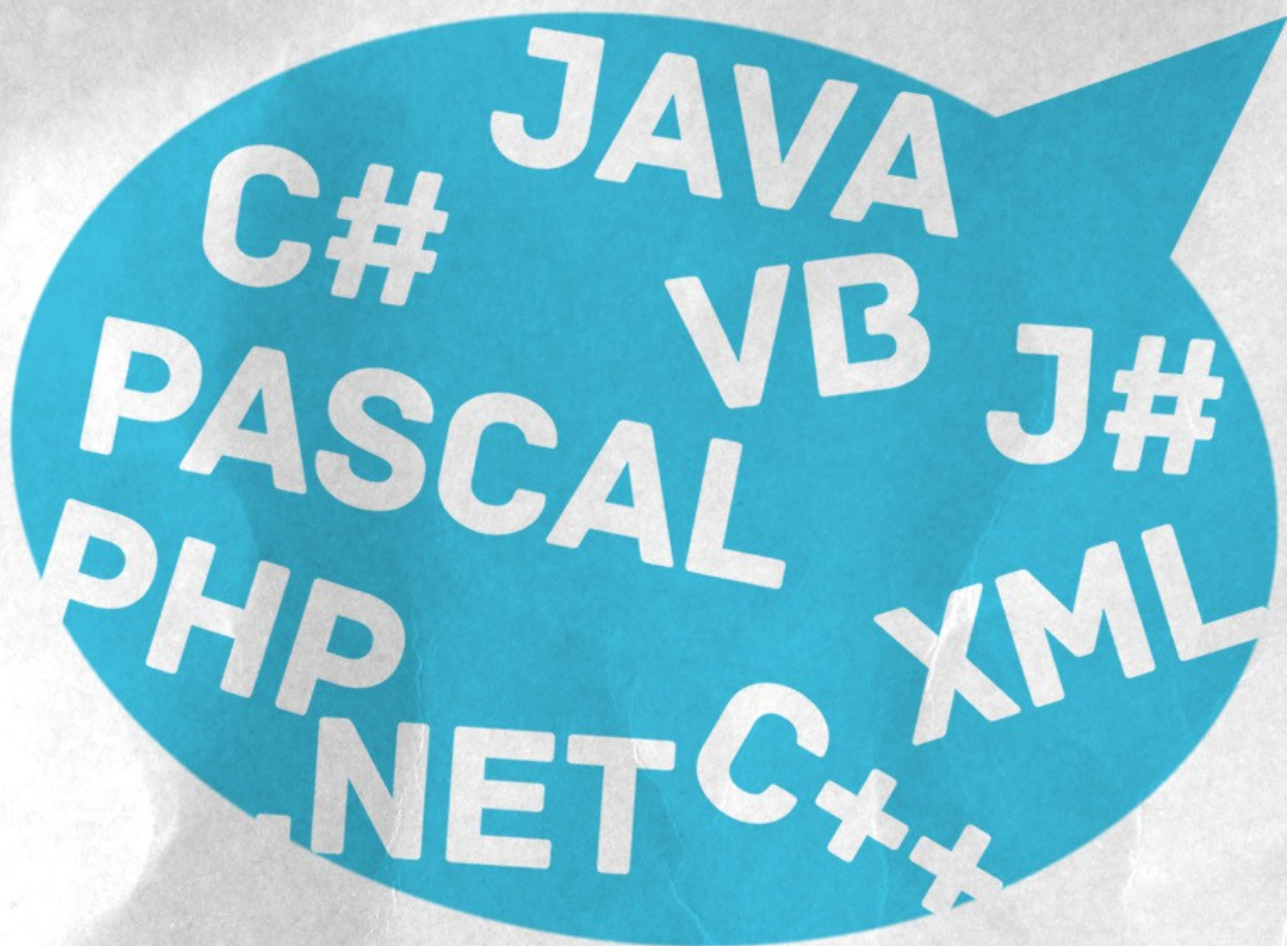
É necessariamente o conhecimento que fará o programador.

AUTOR



Escrito por Telmo Vaz

Estudante de Gestão e Programação de Sistemas Informáticos na instituição EPAD, Grupo Lusófona. Aluno de José Lino e Rui Penacho, dedicou-se ao estudo intensivo de C/C++ e JAVA num curso orientado à informática, com foco no desenvolvimento de aplicações Android.



ENTÃO, SÓ FALAS
EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!



portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE
PROGRAMAÇÃO, APARECE!

COLUMNAS

C# - O GeoCoordinateWatcher como um Serviço Reativo

C# - O GEOCOORDINATEWATCHER COMO UM SERVIÇO REATIVO

Com **Rx**, eventos são cidadãos de primeira classe que podem ser componíveis e passados a outros objetos de forma muito simples.

Implementando O GeoCoordinateWatcher Como Um Serviço Reativo

“[As Extensões Reativas \(Rx\)](#) são uma biblioteca para composição de programas assíncronos e baseados em eventos usando sequências observáveis e operadores estilo-LINQ. Usando Rx, os desenvolvedores **representam** correntes assíncronas de dados com **Observáveis**, **consultam (query)** correntes assíncronas de dados usando **operadores LINQ** e **parametrizam** a concorrência das correntes de dados assíncronas usando **agendadores (Schedulers)**. Pondo de forma simples, Rx = Observables + LINQ + Schedulers.” – da página da [MSDN](#).

A biblioteca providencia também um considerável número de auxiliares que facilitam encapsular eventos em observáveis.

Encapsular o [GeoCoordinateWatcher](#) como um serviço reativo é muito simples. Tudo o que é necessário é expor os eventos como observáveis:

```
public class GeoCoordinateReactiveService :
    IGeoCoordinateReactiveService, IDisposable
{
    private readonly GeoCoordinateWatcher
    geoCoordinateWatcher = new GeoCoordinateWatcher();

    public GeoCoordinateReactiveService()
    {
        this.StatusObservable = Observable
            .FromEventPattern<
                GeoPositionStatusChangedEventArgs>(
                handler => geoCoordinateWatcher.StatusChanged
                    += handler, handler =>
                    geoCoordinateWatcher.StatusChanged -= handler);

        this.PositionObservable = Observable
            .FromEventPattern<GeoPositionChangedEventArgs<
                GeoCoordinate>>(
                handler => geoCoordinateWatcher
                    .PositionChanged += handler,
                handler => geoCoordinateWatch-
                    er.PositionChanged -= handler);
    }

    public IObservable<
        EventPattern<GeoPositionStatus> StatusObservable
    { get; private set; }
}
```

E assim, em vez dos eventos [StatusChanged](#) e [Position-Changed](#) temos respectivamente as correntes de instâncias de [EventPattern<TEventArgs>](#) **StatusObservable** e **PositionObservable**.

Mas a classe **EventPattern<TEventArgs>** inclui a fonte do evento e os seus argumentos em propriedades que são demasiado para as nossas necessidades. Usando normais operadores [LINQ](#) podemos converter as correntes de instân-

cias de **EventPattern<TEventArgs>** em correntes dos valores pretendidos.

```
public class GeoCoordinateReactiveService :
    IGeoCoordinateReactiveService, IDisposable
{
    private readonly GeoCoordinateWatcher
    geoCoordinateWatcher = new GeoCoordinateWatcher();

    public GeoCoordinateReactiveService()
    {
        this.StatusObservable = Observable
            .FromEventPattern<
                GeoPositionStatusChangedEventArgs>(
                handler => geoCoordinateWatcher.StatusChanged
                    += handler, handler =>
                    geoCoordinateWatcher.StatusChanged -= handler)
            .Select(ep => ep.EventArgs.Status);

        this.PositionObservable = Observable
            .FromEventPattern<GeoPositionChangedEventArgs<
                GeoCoordinate>>(handler =>
                geoCoordinateWatcher.PositionChanged += handler,
                handler => geoCoordinateWatcher.PositionChanged
                    -= handler)
            .Select(ep => ep.EventArgs.Position);
    }

    public IObservable<GeoPositionStatus>
        StatusObservable { get; private set; }

    public IObservable<GeoPosition<GeoCoordinate>>
        PositionObservable { get; private set; }
}
```

Usando O GeoCoordinateReactiveService

Tendo criado o envelope [Rx \(wrapper\)](#) para o [GeoCoordinateWatcher](#) demonstrarei agora como pode ser usado numa aplicação simples.

A aplicação apresentará o estado do serviço, a posição e a distância percorrida.

Nesta aplicação simples o serviço será exposto através de uma propriedade *singleton* da classe **App**:

```
public partial class App : Application
{
    // ...

    public static IGeoCoordinateReactiveService
        GeoCoordinateService { get; private set; }

    public App()
    {
        // ...

        InitializePhoneApplication();

        // ...
    }

    // ...

    private void InitializePhoneApplication()
    {
```

```
// ...
GeoCoordinateService = new GeoCoordinate-
ReactiveService();
// ...
}
// ...
}
```

Obter o estado do serviço é muito simples. Apenas requerer subscrever o **StatusObservable**. Uma vez que pretendemos apresentar o estado, precisamos de observá-lo no *dispatcher*:

```
App.GeoCoordinateService.StatusObservable
.ObserveOnDispatcher()
.Subscribe(this.OnStatusChanged);
```

Para a posição fazemos o mesmo com o **PositionObservable**:

```
App.GeoCoordinateService.PositionObservable
.ObserveOnDispatcher()
.Subscribe(this.OnPositionChanged);
```

Obter a distância percorrida parece ser mais complicado porque temos de manter registo da posição anterior e calcular a distância percorrida cada vez que a posição muda. Mas é aqui que a **Rx** mostra as suas vantagens com os seus operadores. Se combinarmos a o observável da posição com o observável da posição [saltando](#) uma posição usando [operador zip](#) obtemos um observável com a posição anterior e a atual. E se aplicarmos um seletor, obtemos apenas a distância percorrida:

```
App.GeoCoordinateService.PositionObservable
.Zip(
    App.GeoCoordinateService.PositionObservable.
    Skip(1), (p1, p2) => p1.Location.GetDistanceTo
    (p2.Location))
.ObserveOnDispatcher()
.Subscribe(this.OnDistanceChanged);
```

Podem encontrar a implementação completa da aplicação [aqui](#).

Recursos

[O GeoCoordinateWatcher Como Serviço Reativo na Galeria MSDN](#)

“ **As Extensões Reativas (Rx) são uma biblioteca para composição de programas assíncronos e baseados em eventos usando sequências** ”

[The Reactive Extensions \(Rx\)... on MSDN](#)

[Rx \(Reactive Extensions\) on CodePlex](#)

- NuGet Packages

[Reactive Extensions - Interfaces Library](#)

[Reactive Extensions - Core Library](#)

[Reactive Extensions - Query Library](#)

[Reactive Extensions - Platform Services Library](#)

[Reactive Extensions - Main Library](#)

[Reactive Extensions - XAML Support Library](#)

[Reactive Extensions - Silverlight Helpers](#)

[Using Rx](#)

[Reactive Extensions \(Rx\) Forum](#)

[Reactive Extensions Team Blog](#)

[MS Open Tech Open Sources Rx \(Reactive Extensions\) – a Cure for Asynchronous Data Streams in Cloud Programming](#)

«Escrito de acordo com novo acordo ortográfico»

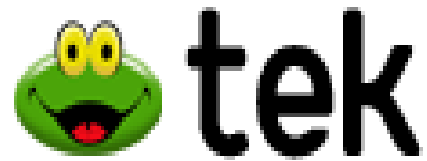
AUTOR



Escrito por Paulo Morgado

É licenciado em Engenharia Electrónica e Telecomunicações (Sistemas Digitais) pelo Instituto Superior de Engenharia de Lisboa e Licenciado em Engenharia Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Pelo seu contributo para a comunidade de desenvolvimento em .NET em língua Portuguesa, a Microsoft premeia-o com o prémio MVP (C#) desde 2003. É ainda co-autor do livro “LINQ Com C#” da FCA.

Media Partners da Revista PROGRAMAR



Análises

Visual Basic 2012: Curso Completo

Multimédia e Tecnologias Interactivas

Visual Basic 2012: Curso Completo

Título: Visual Basic 2012: Curso Completo

Autor: Henrique Loureiro

Editora: FCA - Editora de Informática, Lda.

Páginas: 566

ISBN: 978-972-722-751-8



“O caminho mais curto para conseguir fazer muitas coisas é fazer uma de cada vez”.

Depara-se o leitor com esta citação, assim que abre o livro. Uma frase de Samuel Smiles.

Gosto particularmente da opção de abrir uma obra com uma citação famosa. Confere-lhe alguma mística, uma carga intelectual automática, que felizmente se reflecte nas páginas consequentes. Achei importante a personalidade do prefácio e genial todo o “background” introduzido desde o prólogo aos elementos de programação onde se optou por percorrer as transversalidades da programação informática, sempre de perto com o Visual Basic.

A leitura deste “background” dá uma perfeita noção de incremento, onde a introdução de cada elemento é dada não cedo demais, não tarde demais, mas sempre no momento certo. Para além disso, a linguagem é muito acessível e concisa, o que faz da leitura uma experiência agradável quer para iniciantes, quer para profissionais.

O final de cada capítulo é acompanhado por um bom resumo e um conjunto de perguntas que permitem que o leitor possa ir desafiando o que vai assimilando. As ilustrações e capturas de ecrã são de boa qualidade e usadas na dose certa consoante a necessidade. A ajudar, toda a estrutura da informação está pensada para ser clara e fácil de processar.

Entendo a razão porque o autor optou por focar a relação LINQ com bases de dados mas senti falta de uma referência significativa à sua capacidade de interrogar também colecções. NET, da mesma forma que senti a falta de se falar em LINQ com a classe XDocument, que faz mais sentido no framework alvo da versão 2012 do Studio.

Ainda sobre os elementos que senti falta, a secção que introduz programação orientada por objectos deveria mencionar e enfatizar a capacidade do Visual Basic para organizar classes, com noções de herança e protecção.

Menos importante, mas cada vez mais importante, deveria ter sido abordado o tema da documentação de código em Visual Basic com XML (XML Comments).

Não obstante, o teor do que é apresentado é de elevada qualidade e bastante claro.

Seria sem dúvida complicado chegar a todos os pontos quando por eles quase seria necessário um livro.

É no capítulo do desenvolvimento para Windows 8, e um pouco no de WPF, (ambos capítulos novos, em conjunto com o capítulo de XML) que mais se nota a necessidade de expandir o que ali está concentrado, mas faz o seu trabalho com distinção, como complemento, lembrando que se trata de um livro sobre Visual Basic.

No final, o livro conta com três projectos acompanhados, bastante úteis para colocar em prática inúmeros aspectos da linguagem.

Trata-se de um gestor de uma clínica médica, obtenção e tratamento de dados estatísticos e um “Video Poker”.

O livro não cobre desenvolvimento no Windows Phone, pois pouco se justifica com a adição dos capítulos de WPF e do próprio Windows 8, e não trata de possibilidades com Visual Basic, como ASP.NET, pois não é o verdadeiro intuito.

Em suma, **Visual Basic 2012: Curso Completo** enfatiza realmente a expressão **Curso Completo**, e considero uma excelente base não só para quem quer aprender Visual Basic no Visual Studio 2012, mas também para quem quer aprender Visual Basic, ponto.

Considero tratar-se de uma excelente companhia para iniciantes, e até mesmo para profissionais que se esqueceram de um ou outro pormenor de sintaxe, principalmente no capítulo de WPF dado a sua natureza concentrada, mas variada.

AUTOR



Escreito por Sérgio Ribeiro

Curioso e autodidata com uma enorme paixão por tecnologias de informação e uma saudável relação com a .NET Framework. Moderador global na comunidade Portugal@Programar desde Setembro de 2009. Alguns frutos do seu trabalho podem ser encontrados em <http://www.sergioribeiro.com>

Multimédia e Tecnologias Interactivas

Título: Multimédia e Tecnologias Interativas (5ª Edição Atualizada e Aumentada)

Autor: Nuno Magalhães Ribeiro

Editora: FCA

Páginas: 664

ISBN: 978-972-722-744-0



Um tema sempre atual e do qual nenhum estudante ou empresa do ramo tecnológico consegue estar dissociado. Uma área em constante evolução, onde os conhecimentos teóricos e práticos são as melhores ferramentas que se podem ter para desenvolver novas ideias e projetos inovadores. Numa primeira análise, facilmente concluímos que se trata de um livro académico, suportado pelas palavras do autor, que refere "(...) o livro foi sendo utilizado, aprofundado, verificado e testado no contexto do ensino superior (...)"; contudo, também poderá ser usado na componente de auto-aprendizagem, pois é bastante claro na explicação de vários temas.

Os temas e conceitos são diversos. Multimédia e Tecnologias Multimédia - o que é, para que servem, onde são utilizados. Explica os vários tipos de multimédia e os objetivos de cada tipo. Uma excelente exposição teórica dos fundamentos da Multimédia. Representação Digital e Interatividade - como funcionam as representações digitais, quais os níveis de interatividade destas representações e variedade de meios. Contém alguns conceitos interessantes e práticos como a conversão de sistemas de numeração binária e digital. Aplicações Multimédia Interativas para educação, empresas e público em geral. Fala da classificação das diferentes aplicações multimédia, apresentando vários exemplos em cada área, ajudando a perceber as diferentes aplicações e tecnologias. Tipos de média estáticos (imagens, texto e vetorial) e dinâmicos (vídeo, animação, áudio). Como funcionam, como são codificados, resoluções, modelos de cor e compressão. Explicam bem as diferenças entre cada um.

E autoria Multimédia, combinação e organização de conteúdos dos vários tipos de media. Aborda outras temáticas como a gestão e desenvolvimento de projetos multimédia, mencionando e descrevendo as suas várias fases, quais as melhores técnicas a adotar e como devem ser executadas. Passagem ideal para o desenvolvimento de projetos, inclui-

do no último capítulo do livro. Componente teórica sobre os suportes óticos para multimédia. Quais são, vantagens e desvantagens, quais as características e como funcionam. Inclui informação da codificação utilizada. Componente Web, onde são abordadas as suas várias linguagens, o desenvolvimento de aplicações, o seu método de funcionamento, explicando protocolos, as linguagens utilizadas, e estruturas. Foca também o tipo de aplicações que existem, não esquecendo referências a HTML5 e Web 3.0. Sistemas Multimédia, com sistemas de realidade virtual, televisão digital e interativa, e em dispositivos móveis; como rapidamente se tornaram fiéis e indispensáveis ao nosso dia-a-dia e como ajudaram a melhorar a nossa qualidade de vida.

O último capítulo poderá, sem dúvida, ser ajuda para docentes e alunos nas diferentes áreas de Tecnologias Multimédia, com várias propostas de projetos práticos com objetivos claros e que serão bastante úteis para a utilização de conceitos e tecnologias documentadas no livro. Sendo um livro académico, contém muitos conceitos teóricos, definições e introduções. Não será um livro utilizado por empresas, mas sim por pessoas que estão a "entrar" pela primeira vez na área do desenvolvimento de aplicações multimédia. Permite ter a noção de suportes, tecnologias existentes e as suas potencialidades, mas não transforma os leitores em "experts" imediatos, pois só com tempo e trabalho isso é possível. A componente teórica está bem presente, correspondendo ao objetivo primordial do livro. Um pouco mais de 600 páginas de fácil leitura e interpretação, com exemplos práticos e úteis. Um dos pontos fortes, é sem dúvida, o resumo e exercícios propostos no final de cada capítulo, que ajudam a perceber melhor os conceitos e a assimilar conhecimentos.

O aconselhável será a inclusão de mais exemplos práticos, e porque não, tratando-se de um livro académico, a inclusão de exemplos de trabalhos realizados por alunos. Um livro que se pode dizer ser mais do que um simples manual, que aconselhamos para disciplinas com componente teórica e prática na área de Multimédia e Tecnologias Interativas.

Dar também claro os parabéns ao autor Nuno Ribeiro, que para além de um trabalho enorme de investigação e pesquisa, conjugou toda a informação de uma forma prática e simples, notando-se a sua preocupação por deixar claros todos os conteúdos do livro.

AUTOR



Escrito por Sérgio Laranjeira

Sonhador, empreendedor, idealista e apaixonado. Coloca em cada projeto todas as suas virtudes humanas, fundindo-as com mestria nas suas skills enquanto programador. É na área web que revela grande conhecimento e uma capacidade técnica pragmática. Licenciado em Computação Gráfica, é apaixonado pelo mundo dos 3 w's e por tudo o que está relacionado com usabilidade e acessibilidade. Transmite toda a sua paixão pela Web e Multimédia nas diferentes disciplinas que leciona no Instituto Politécnico de Viana do Castelo.

COMUNIDADES

Comunidade NetPonto – Certificação Microsoft

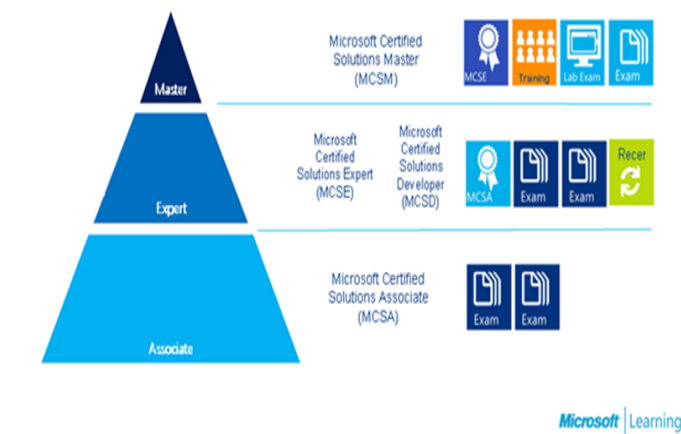
Certificação Microsoft

Na edição nº29 da Revista Programar, foi lançado um artigo sobre Certificação Microsoft®, este artigo teve como objetivo dar a conhecer o que é a Certificação Microsoft®, qual o percurso a percorrer para concretizar a certificação e quais os benefícios. Nesta edição, pretendo apresentar uma atualização sobre o percurso e apresentação de novos títulos.

É mais que sabido que a tecnologia está em constante mudança e a pare disso a Microsoft® tem planeado novas estratégias ao nível da certificação. Estratégias essas que vão desde apresentação de novos percursos, assim como a “exigência” de atualização do conteúdo das mesmas (refirme ao fato de haver algumas certificações que requerem a atualização da certificação a cada dois anos).

Atualmente diria que estamos numa fase de transição entre a nova certificação e a antiga certificação, isto é, até ao dia 31 de Julho de 2013 ainda vai ser possível obter a certificação que falei no artigo da edição nº29, mas a partir dessa data, a maior parte dos exames deixará de estar disponível. Para mais informações sobre lista de certificações retiradas consultem <http://bit.ly/VS1z1m> e para saberem mais sobre quais exames que ainda estão disponíveis até à data mencionada, consultem “Others certifications” em <http://bit.ly/X1boXJ> e em “Upgrade paths”, da mesma referência, o tópico sobre “MCPD on Visual Studio 2008 certification to MCPD on Visual Studio 2010”.

Na nova certificação, a Microsoft introduziu uma nova hierarquia:



Como é possível ver na imagem, temos então os seguintes títulos:

- Microsoft Certified Solutions Master (MCSM)

- Microsoft Certified Solutions Expert (MCSE)
- Microsoft Certified Solutions Developer (MCSD)
- Microsoft Certified Solutions Associate (MCSA)

Nota: Estes títulos não estão disponíveis em todas as categorias (que são: Server, Desktop, Database e Developer). Podem constatar isso nesta referência: “Certification Overview” - <http://bit.ly/13pWmAj>.

Na categoria de Developer vamos ter os títulos:



E as especializações:

- Programming in HTML5 with JavaScript and CSS3 (Exame 70-480: <http://bit.ly/13q0Arl>)
- Specialist Programming in C# Specialist (Exame 70-483: <http://bit.ly/ZTE5MN>)

Para obtenção do **MCSD: Windows Store Apps** podemos escolher as seguintes opções:

- **MCSD: Windows Store Apps Using HTML5**
 - Título que requer aprovação nos exames:
 - Programming in HTML5 with JavaScript and CSS3 (Exame 70-480: <http://bit.ly/13q0Arl>)
 - Essentials of Developing Windows Store Apps Using HTML5 and JavaScript (Exame 70-481: <http://bit.ly/WA5hsP>)
 - Advanced Windows Store App Development Using HTML5 and JavaScript (Exame 70-482: <http://bit.ly/V2YdrE>)

CERTIFICAÇÃO MICROSOFT

• **MCSA: Windows Store Apps Using C#**

- Título que requer aprovação nos exames:
- Programming in C# (Exame 70-483: <http://bit.ly/ZTE5MN>)
- Essentials of Developing Windows Store Apps Using C# (Exame 70-484: <http://bit.ly/RM5M6j>)
- Advanced Windows Store App Development Using C# (Exame 70-485: <http://bit.ly/WNrdjX>)

Para obtenção do **MCSA: Web Applications** é necessário aprovação nos exames:

- Programming in HTML5 with JavaScript and CSS3 (Exame 70-480: <http://bit.ly/13q0Arl>)
- Developing ASP.NET 4.5 MVC Web Applications (Exame 70-486: <http://bit.ly/WA5Nag>)
- Developing Windows Azure and Web Services (Exame 70-487: <http://bit.ly/ZH28tm>)

Para obtenção do **MCSA: Application Lifecycle Management** é necessário aprovação nos exames:

- Administering Microsoft Visual Studio Team Foundation Server 2012 (Exame 70-496: <http://bit.ly/SqIOU0>)
- Software Testing with Visual Studio 2012 (Exame 70-497 <http://bit.ly/UGNlrw>)
- Delivering Continuous Value with Visual Studio 2012 Application Lifecycle Management (Exame 70-498: <http://bit.ly/ZTPLiA>)

Para obtenção do **MCPD: Windows Phone Developer*** é necessário aprovação nos exames:

- Silverlight 4, Development (Exame 70-506**: <http://bit.ly/WA6rEW>)
- Accessing Data with Microsoft .NET Framework 4 (Exame 70-516: <http://bit.ly/ZTHuLp>)
- PRO: Designing and Developing Windows Phone Applications (Exame 70-599: <http://bit.ly/ZTHyuA>)

* Requer que seja atualizada a cada dois anos.

** Exames será retirado a 31 de Julho de 2013.

Nota: Com o lançamento do Windows Phone 8 SDK acredito que em breve seja publicado mais novidades sobre este título.

Uma última nota relativamente aos detentores de certificação Microsoft, estes não perdem os títulos obtido, pois essa certificação não tem validade, apenas é recomendável que façam uma atualização da mesma para que a sua certificação se mantenha atual.

Um caso concreto: quem obteve o MCPD: Web

“ **A certificação Microsoft® continua a ser uma boa aposta na carreira profissional de um TI** ”

Developer 4 deverá fazer uma atualização para MCSA: Web Applications, mas atenção não é necessário fazer os exames todos novamente, apenas é preciso fazer um exame específico “upgrade” e em alguns casos fazer mais um exame específico, isto é, tem que passar com sucesso nos exames:

- Programming in HTML5 with JavaScript and CSS3 (70-480)
- Upgrade your MCPD: Web Developer 4 to MCSA: Web Applications (70-492)

Relativo a esta certificação, deixo algumas motivações para concretizar a certificação:

Cursos gratuitos:

- Developing in HTML5 with JavaScript and CSS3 Jump Start em <http://bit.ly/WTAfi3>.
- Developing Windows Store Apps with HTML5 Jump Start em <http://bit.ly/13oBBG8>

Exame gratuito

- Obtém o exame Programming in HTML5 with JavaScript and CSS3 (Exame 70-480: <http://bit.ly/13q0ArI>) gratuitamente! Código do voucher é HTMLJMP e é válido até 31 de Março de 2013.

“ **Atualmente diria que estamos numa fase de transição entre a nova certificação e a antiga certificação** ”

Segunda oportunidade (“Second Shot”)

- Obtém uma segunda oportunidade para realizar o exame que te vais submeter aprovação, consulta em <http://bit.ly/VScx7b>

Descontos de 15% em pacotes de exames:

- Para os interessados em obter o MCSD: Windows Store Apps existe um desconto de 15% para o pacotes dos vários exames, consulta em <http://bit.ly/SqH25k>

Nota:

- 1. Voucher válidos até dia 31 de Dezembro de 2013.
- 2. É acumulável com a oferta de “**Second Shot**”.
- 3. Para saber em quanto vai ficar, consulta em <http://bit.ly/WA9Cw4> e escolhe Portugal!

Para terminar, saliento ainda os títulos relacionados com SQL Server e SharePoint:

Na categoria de Database vamos ter vários títulos:

- MCSA: SQL Server disponível em <http://bit.ly/UUfa6S>
- MCSE: Data Platform disponível em <http://bit.ly/VTXyY4>
- MCSE: Business Intelligence disponível em <http://bit.ly/VTXyY4>
- MCSM: Data Platform disponível em <http://bit.ly/WNVFul>

Na categoria Server, podemos encontrar os títulos relacionados com SharePoint, temos então:

- MCSE: SharePoint informação disponível em <http://bit.ly/WNFBZD>
- MCSM: SharePoint informação disponível em <http://bit.ly/V34vYn>

Em conclusão, a certificação Microsoft® continua a ser uma boa aposta na carreira profissional de um TI, pois as atualizações disponíveis permitem um maior desenvolvimento técnico nas diversas tecnologias.

Referência oficial: <http://www.microsoft.com/learning>.

«Escrito de de acordo com o novo acordo ortográfico»

AUTOR



Escrito Por Sara Silva

é mais recente Microsoft MVP em Portugal. De formação base, licenciatura em Matemática – Especialidade em Computação, pela Universidade de Coimbra, é também Microsoft Certified Professional Developer. Atualmente o seu foco de desenvolvimento incide em Windows Phone e Windows 8 Store Apps.

No Code

Programar—como começar?

Novidades na Pesquisa no SharePoint 2013

SEO – Acessibilidades, Usabilidade e Demografia

Entrevista A Miguel Gonçalves

Projecto em Destaque na Comunidade Portugal –a– Programar : PROGRAMAR

Programar—como começar?

O leitor poderá achar estranho (sendo já um programador) ao ler um artigo sobre um aspecto *passado* do seu percurso. Sob esse ponto de vista, este artigo poderá até parecer *redundante*, mas penso que o tema em questão justifica bem a “redundância”, tendo em conta que a programação é um exercício cada vez mais acessível a qualquer pessoa.

Existe na comunidade de programadores uma estranha ideia de que toda gente deve começar a programar com a linguagem X, Y ou Z. Frequentemente (e como se pode ver em discussões acesas no), as linguagens X, Y e Z são frequentemente a linguagem C. Chegámos mesmo ao cúmulo de ver afirmações que parecem sugerir que começar com uma linguagem dinâmica de alto nível (quase diametralmente oposta a C) é equivalente a uma lobotomia seguida da incapacidade permanente de escrever programas eficientes, de baixo nível.

Eu comecei a programar durante o ensino básico (com 12–13 anos) e fi-lo essencialmente sem ajuda de ninguém. Comecei com uma variante de Pascal (no Delphi 1.0), e com ela permaneci durante alguns anos. Embora a minha ideia de programa fosse nessa altura algo “quadrada” (com uma barra azul e três botões cinzentos acima, à direita), lá consegui compreender que a *essência* de programar não era um conjunto de janelas e botões bonitos. Não, a essência de programar é resolver ou simplificar problemas.

Entretanto fui aprendendo outras linguagens (continuo a fazê-lo), mas vou apenas partilhar a minha primeira impressão de C, vindo de Pascal: C tinha uma sintaxe absurda. Não me refiro à questão das chavetas vs begins e ends mas sim à ambiguidade da sintaxe. Em C, * pode ser várias coisas (o mesmo para &), e até mesmo as declarações de variáveis são idênticas aos protótipos de funções. Neste aspecto, Pascal pareceu-me uma linguagem muito mais regrada, pensada e coerente. A sua sintaxe (bastante mais explícita que a de C) exige mais dos teclados, mas bons editores de texto e ambientes de desenvolvimento automatizam algumas partes da escrita do código, suavizando este “problema”.

Deparei-me também com problemas que não tinha encontrado em Pascal: em C não havia um tipo de dados string como em Pascal.^[1] As strings de Pascal têm as suas desvantagens, mas para pequenos programas são bastante razoáveis e muito mais confortáveis que as “strings” de C.^[2] Também senti falta da flexibilidade do sistema de tipos de dados de Pascal quando comecei a utilizar C.

Depois havia também o problema da modularização de código em C. Vindo de Pascal, estava habituado às *units* de

código com secções explícitas para a *interface* e *implementação*. Em C fui obrigado a aprender as artimanhas dos *#includes* e *include guards*.^[3]

Até na questão da gestão de memória e apontadores Pascal me oferecia uma sintaxe mais lógica que C.

Antes que me desvie mais do tema principal, vou resumir: senti-me um pouco perplexo acerca dos motivos pelos quais esta linguagem aparentemente primitiva, confusa e mal pensada era a mais utilizada em todo o mundo. Eventualmente lá compreendi que “a ideia” da linguagem C era ser compacta, concisa, desprovida de redundâncias. Hoje em dia aprecio imenso essas características, e a sintaxe não me incomoda mais (mas ainda a acho ambígua). Ainda assim, fica-me a ideia de que esta linguagem não é a mais indicada para quem está a começar a programar devido aos problemas que falei acima e outros como contacto precoce e “demasiado intenso” com gestão manual de memória, apontadores, entre outros.

“ **Sem mutabilidade, ciclos e outros “artifícios”, as linguagens funcionais obrigam o programador a pensar nos problemas de outra forma** ”

Esta “sub-secção” do artigo não foi escrita com o objectivo de comparar Pascal com C (isso seria tema suficiente para um artigo por si só), mas sim com o objectivo de mostrar problemas que eu enfrentei quando aprendi C e que provavelmente toda a gente enfrenta quando começa a programar em C. É importante reter que eu já tinha experiência com Pascal (e Javascript, entre outras) e mesmo assim encontrei os “defeitos” acima referidos.

No Code

PROGRAMAR—COMO COMEÇAR?

Peguemos agora no problema por outra ponta. Que linguagem devemos incentivar o jovem programador a escolher como sua primeira aposta? É uma questão complexa, e a resposta não é mais simples, mas penso que podemos dar algumas aproximações.

Na *minha* opinião, C não é a melhor escolha para primeira linguagem. Pascal (já que já falámos dele) é uma opção bem melhor e menos desmotivante para quem não tem ideia alguma do que é programar^[4], mas não precisamos de ficar com Pascal. Podemos pegar, por exemplo, em Python. Python, sendo uma linguagem dinâmica nos tipos de dados e de alto nível de abstracção, consegue ser bastante expressiva e, por esse motivo, adequada a quem se inicia na programação precisamente por facilitar a transposição das ideias para código^[5]. Ainda por cima, Python oferece suporte para outros paradigmas de programação (objectivo, funcional) sendo por isso uma “plataforma” a partir da qual o programador pode partir para novos desafios.

Mas deixemos o Python de lado por agora. Existem muitas outras linguagens que podem preencher o seu lugar. Ruby, por exemplo, é praticamente equivalente a Python, com algumas diferenças sintácticas e operacionais, mas essencialmente idêntica. Este tipo de linguagens oferecem uma prototipagem rápida, um nível de expressividade e abstracção elevados e por isso devemos encorajar os iniciados a brincar com elas desde cedo.

Consideremos agora o paradigma funcional. São estas as linguagens que deveríamos aprender primeiro. Com uma boa dose de inspiração da matemática, estas linguagens dão grande ênfase à composição de funções e à transparência referencial.^[6]

Sem mutabilidade, ciclos e outros “artifícios”, as linguagens funcionais obrigam o programador a pensar nos problemas de outra forma e mesmo que na sua vida profissional não venham a programar numa destas linguagens, serão melhores programadores simplesmente pelo contacto prévio com este paradigma.

Existem diversas opções neste ramo, mas vou salientar apenas 2: Haskell e Scheme (um dialecto de Lisp).

Haskell é uma linguagem funcional pura, com um sistema de tipos de dados impressionante. O conjunto de operações que fornece para manipularmos funções é simplesmente incrível e torna-se muito difícil tentar descrever a um programador de C (ou Python, ou outra linguagem imperativa) o que é possível fazer em Haskell, mas vale a pena aprender. Com o seu sistema de tipos de dados, os programas em Haskell “surgem” muitas vezes como consequência directa dos dados envolvidos nos problemas que tentamos resolver. Além disso, com Haskell o programador é obrigado a pensar em termos de recursividade e ausência de alterações de estado, coisas que se não forem aprendidas cedo, mais

difícilmente se enraizarão no futuro. Funções de ordem superior, aplicação parcial de funções, *closures*, funções anónimas (*lambdas*) são apenas alguns dos conceitos ubíquos em Haskell e praticamente ausentes de outras linguagens não-funcionais que só serão utilizados por quem os conhece. Em Haskell temos também um impressionante sistema de tipos de dados que não vou tentar explicar aqui; só mesmo experimentando se pode ter a noção do quão avançado e flexível ele é.

É esse um dos problemas de começar por uma linguagem de baixo nível: como tudo eventualmente é compilado para código binário, como todas as abstracções de alto nível são reduzidas a operações de baixo nível, é pouco provável que um programador tome a iniciativa de investigar estas operações de alto nível quando já consegue fazer “tudo o que precisa” em termos de operações de baixo nível (ainda que o código seja maior, mais fálvel e mais difícil de manter).

“ **Programar é muito mais que compreender o processador, Programar é saber pensar, é saber criar, Programar é inovar.** ”

Desviei-me novamente. Falemos agora de Scheme. Em Scheme, o jovem programador encontra uma linguagem de sintaxe *extremamente simples* (não necessariamente agradável, mas é uma questão de hábito) e que possibilita programar em *vários paradigmas!* Ou seja, com Scheme podemos não só escrever código parecido com C (puramente imperativo, variáveis globais, mutabilidade), como podemos rapidamente alternar para o paradigma funcional. Essencialmente, podemos pensar em Scheme como o melhor de dois mundos: temos a possibilidade de programar de forma puramente funcional ou puramente imperativa.

É também possível em Scheme fazer algo denominado *meta-programação*^[7], permitindo programar o compilador (ou interpretador) para escrever parte do nosso programa por nós; podemos efectivamente adicionar nova sintaxe à linguagem!, e adequá-la aos problemas que tentamos resolver^[8].

Após esta breve análise de possíveis primeiras linguagens de programação, resta-me concluir com aquilo que penso ser o nosso dever como programadores (e possíveis mentores): devemos encorajar um pensamento aberto, de alto nível, de forma a que os novos programadores não fiquem presos ao passado do baixo nível. Obviamente, não devemos descurar a preocupação com a eficiência dos programas que fazemos, mas não devemos sacrificar os horizontes intelectuais de quem está a aprender a favor de um algoritmo uns milissegundos mais rápido. Este é um (des-)equilíbrio que tem estado demasiado virado para a eficiência, relegando para segundo plano a experimentação e pensamento criativo.

“ Não devemos descurar a preocupação com a eficiência dos programas que fazemos, mas não devemos sacrificar os horizontes intelectuais de quem está a aprender a favor de um algoritmo uns milissegundos mais rápido. ”

É preciso combater a ideia de que a aprendizagem deve ser focada unicamente no mercado. É preciso dar a quem

aprende a liberdade de utilizar linguagens conceptualmente mais ricas, mesmo que a sua aplicabilidade no mercado seja baixa (ou até nula). É preciso compreender que *aprender* a programar não é um emprego mas sim uma plataforma para actividades futuras; é a base de todo o conhecimento de um programador, e por isso não devemos aceitar que se reduza a questão a um simples “C é a linguagem mais usada e ficas a saber como funciona o computador, por isso é por essa que deves começar”.

Programar é muito mais que compreender o processador. Programar é saber pensar, é saber criar. Programar é inovar.

Transmitamos isto a quem começa :)

Notas:

- ¹ Naturalmente, a representação de uma string em Pascal é menos eficiente em termos de memória, mas apresenta alguns ganhos de eficiência nalgumas situações. De qualquer modo, as tradicionais “strings de C” também estão disponíveis em Pascal sob o nome PChar.
- ² Na realidade, não existem strings em C; o que existe é um conjunto de funções que lida com arrays do tipo char[].
- ³ Isto acontece porque C não tem verdadeiro suporte para modularização de código.
- ⁴ Não subestimemos o valor da motivação na (auto-)aprendizagem.
- ⁵ Python pode até mesmo ser considerado “pseudo-código executável”, de certa forma.
- ⁶ Ou seja, funções que têm sempre o mesmo resultado para o mesmo conjunto de argumentos.
- ⁷ Este tipo de meta-programação é incomparavelmente melhor que as tradicionais macros de C/C++ ou os templates de C++.
- ⁸ Mais uma vez, não comparemos as macros de C/C++ com isto, são coisas completamente diferentes.

AUTOR



Escrito por António Pedro Cunha (pwseo)

Médico natural de Guimarães, formado na Universidade do Minho.

Programador autodidacta em parte dos tempos livres, inscrito no fórum desde 2006.

Website: <http://pwseo.aloj.net>



XX SINFO

A **Semana Informática (SINFO)** do IST (Instituto Superior Técnico) é um evento inteiramente organizado pelos alunos de Engenharia Informática e realiza-se no Pavilhão de Civil do *campus* Alameda, em Lisboa, de **25 de Fevereiro a 1 de Março de 2013**. A **entrada é livre e gratuita** para todos os interessados em Tecnologias de Informação.

Oradores Convidados

Rob Bishop - developer e evangelista da Fundação Raspberry-Pi

Ray Muzyka - fundador da Bioware

Joel Spolsky - fundador do Stack Overflow

Colin Johanson - lead designer na Arena-Net

Afonso Salcedo - lightning artist na Dreamworks e Pixar

Kevin Warwick - professor de Cibernética na Universidade de Reading

Allan McRae - developer do Arch Linux

Rick Falkvinge - fundador do Partido Pirata Sueco

entre outros



E ainda:

Workshops - Android, Blender, RaspberryPi, Eclipse, Robótica, Game Design e muitos mais

Exposição Tecnológica, Apresentações empresariais, Painéis, etc.

Para mais informações consultar:

Site
<http://www.sinfo.org>

Redes sociais
<https://www.facebook.com/sinfoist>
https://twitter.com/#!/sinfo_ist
<http://www.youtube.com/sinfoist>

Mail
geral@sinfo.org



Novidades na Pesquisa no SharePoint 2013

Introdução

Desde a sua primeira versão que o SharePoint inclui funcionalidades de pesquisa. Nessa altura ainda se tratavam de dois produtos distintos e pouco integrados, o **SharePoint Portal Server 2001** e o **SharePoint Team Services**, mas já era possível definir *Best Bets*, indexar documentos e respetiva meta-informação, incluir no índice conteúdos de *file servers*, *web servers*, Lotus Domino e Exchange Public Folders, e até estender a pesquisa através de *iFilters*. Porque esta pesquisa era implementada sobre os catálogos do IIS e limitada aos documentos armazenados no sistema de ficheiros, não era possível pesquisar itens de listas como tarefas, contactos ou fóruns de discussão.

Com o **SharePoint 2003**, os conteúdos dos *sites* passaram a ser armazenados em bases de dados SQL Server permitindo-lhe tirar partido das funcionalidades de *Full-Text Search* do SQL Server 2000/2005. Nesta versão já era possível pesquisar conteúdos de listas, pessoas, *sites*, documentos e imagens, e definir alertas para resultados de pesquisa. Foi também nesta versão que a plataforma passou a suportar *stemming* para alguns idiomas. Ou seja, os termos pesquisados são expandidos para incluir as respetivas inflexões. Por exemplo, numa pesquisa por “swim” são retornados também resultados para os termos “swum”, “swum” e “swimming”.

O **SharePoint 2007** trouxe um leque alargado de novas funcionalidades e muitas melhorias para a Pesquisa que passou a ser apresentada como um dos seis pilares principais da plataforma, a par de *Collaboration*, *Portal*, *Content Management*, *Business Forms* e *Business Intelligence*. Entre estas a possibilidade de pesquisar pessoas não apenas pelo nome ou cargo, mas também por *expertise*, *social distance* e interesses comuns, e a capacidade de indexar dados armazenados em sistemas externos (*Line of Business Applications*) através do *Business Data Catalog* (BDC).

O investimento na Pesquisa continuou com o **SharePoint 2010**, em particular com a aquisição pela Microsoft da FAST Search & Transfer, a empresa norueguesa criadora da plataforma **FAST ESP**, durante o ciclo de desenvolvimento do produto. Na “tarte” do SharePoint 2010 as “fatias” mudaram de nome, mas a Pesquisa continuou a ser uma delas, a par de *Sites*, *Communities*, *Content*, *Insights* e *Composites*.

O SharePoint 2010 oferece três níveis diferentes de pesquisa:

- **SharePoint Foundation Search** – o nível de entrada,

oferecido com a versão gratuita do produto e com bastantes limitações funcionais e de capacidade (indexa um máximo de 10 milhões de itens).

- **SharePoint Server Search** – o nível intermédio, oferece todas as funcionalidades existentes na versão 2007 e ainda acrescenta novas *web parts*, melhorias na pesquisa de pessoas, melhorias à *Keyword Query Language*, maior capacidade (indexa um máximo de 100 milhões de itens), pesquisa federada, refinamentos básicos (*shallow refinements*), integração com taxonomias e *cross-farm searching*, ou seja, pesquisa partilhada entre *farms*.
- **FAST Search for SharePoint** – o nível superior, corresponde a substituir o motor de pesquisa original do SharePoint pelo FAST, com uma arquitetura mais flexível e uma escalabilidade muito superior (pode indexar mais de 500 milhões de itens).

Recorrendo ao FAST Search for SharePoint 2010, conseguimos ter tudo o que é oferecido pelas versões inferiores da pesquisa e ainda um conjunto impressionante de novas capacidades, entre elas:

- Modelos de *ranking* customizados com *tuning* da relevância
- *Visual Best Bets*
- Particionamento do índice por vários servidores, possibilitando uma escalabilidade muito superior às soluções clássicas
- Refinamentos avançados (*deep refinements*)
- Ordenação de resultados por propriedades (e não apenas por relevância)
- Extração automática de propriedades
- Pesquisa contextual
- Previsão de documentos *Office* diretamente na lista de resultados de pesquisa

No **SharePoint 2013** a Pesquisa é uma componente ainda mais central e crítica da plataforma, suportando muitas das novas funcionalidades do produto. As secções seguintes apresentam as principais novidades do SharePoint 2013 na área da pesquisa.

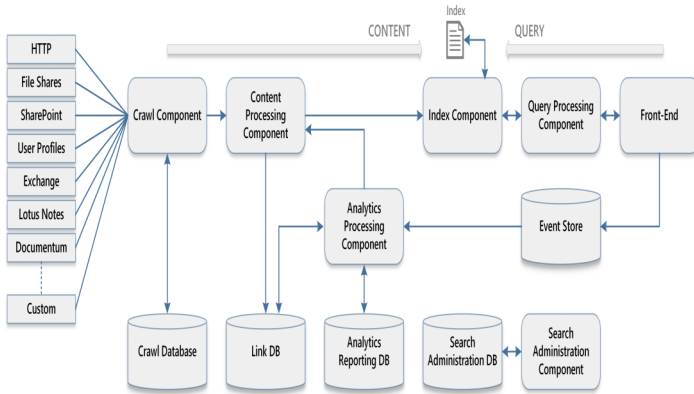
Nova arquitetura lógica

A maior e talvez mais esperada novidade na pesquisa é a

No Code

NOVIDADES NA PESQUISA NO SHAREPOINT 2013

integração completa da tecnologia FAST Search no SharePoint 2013. Isto significa que toda a arquitetura lógica da componente de pesquisa do SharePoint foi redesenhada e é agora muito mais próxima da arquitetura do FAST Search for SharePoint 2010 do que da arquitetura original da pesquisa utilizada nas versões anteriores do produto.



A nova arquitetura herda a extrema escalabilidade do FAST bem como grande parte das suas capacidades de processamento analítico e linguístico, e acrescenta-lhe um conjunto de novas funcionalidades. Os vários componentes estão descritos nas secções seguintes.

Componente de crawl

Também conhecido por *bot*, *spider* ou *crawler*, este componente existe na maioria dos motores de pesquisa. É responsável por percorrer as fontes de conteúdos (*content sources*), recolhendo a informação nelas contida, e por passar essa informação ao componente de processamento de conteúdos (*content processing component*). Este componente não faz qualquer tipo de processamento da informação recolhida.

O componente de *crawl* utiliza uma base de dados, designada por *Crawl Database*, para guardar informação sobre os itens recolhidos e informação sobre o processo de recolha, como a data/hora e o tipo de atualização do último *crawl*. Esta base de dados não guarda a informação recolhida propriamente dita, isso será alvo de processamento e armazenamento pelos outros componentes da arquitetura.

Foram realizadas várias melhorias ao componente de *crawl* nesta nova versão mas a mais interessante é o novo conceito de **continuous crawling**. Trata-se de um novo modo de *crawl* que pode ser aplicado apenas a fontes de conteúdos do tipo SharePoint e que permite que atualizações aos conteúdos fiquem disponíveis para pesquisa muito rapidamente. Na prática, o que acontece é que o *crawler* corre a cada 15 minutos (por omissão, mas é configurável) e consegue recolher apenas as alterações que foram realizadas aos conteúdos nesse período. Como resultado, é possível que um novo documento fique disponível para pesquisa apenas alguns segundos depois de ter sido criado.

Outras das melhorias realizadas sobre este componente é o suporte para **anonymous crawling**, ou seja, o *crawler* percorre os conteúdos como um utilizador anónimo. No passado, o componente de *crawl* tinha que ter uma conta associada para aceder aos conteúdos.

Connectors

Para aceder às várias fontes de conteúdos, o componente de *crawl* utiliza *connectors* (também conhecidos por *protocol handlers*) que são módulos de *software* cujo objetivo é percorrer um tipo específico de fonte de conteúdos e extrair a informação que esta contém. O SharePoint 2013 inclui um conjunto de *connectors*:

- **SharePoint** – permite percorrer um portal baseado em SharePoint e extrair a informação armazenada nas suas páginas, listas e documentos.
- **HTTP** – permite percorrer um portal web, qualquer que seja a sua tecnologia de base, obtendo a informação das suas páginas e navegando pelas cadeias de hiperligações entre as páginas.
- **File Share** – permite percorrer estruturas de pastas num sistema de ficheiros extraindo a informação armazenadas nos ficheiros cujos formatos sejam reconhecidos.
- **People Profile Connector** – permite percorrer os perfis dos utilizadores em SharePoint obtendo a informação que estes armazenam nos seus campos.

Adicionalmente, através da *framework* BDC (*Business Data Connectivity*) podemos criar novos *connectors* que permitam ao SharePoint percorrer a informação armazenada em virtualmente qualquer sistema. O SharePoint 2013 inclui também alguns *connectors* baseados nesta *framework*:

- **Exchange Public Folders** – permite percorrer as *Public Folders* de um servidor *Microsoft Exchange* e obter a informação armazenada nos ficheiros que estas contêm.
- **Lotus Notes** – permite obter a informação armazenada neste sistema.
- **Documentum Connector** – permite percorrer os documentos armazenados neste sistema e obtendo a informação que estes contêm bem como a sua meta-informação.
- **Taxonomy Connector** – permite percorrer as taxonomias criadas no próprio SharePoint e geridas pela *Metadata Management Service Application*.

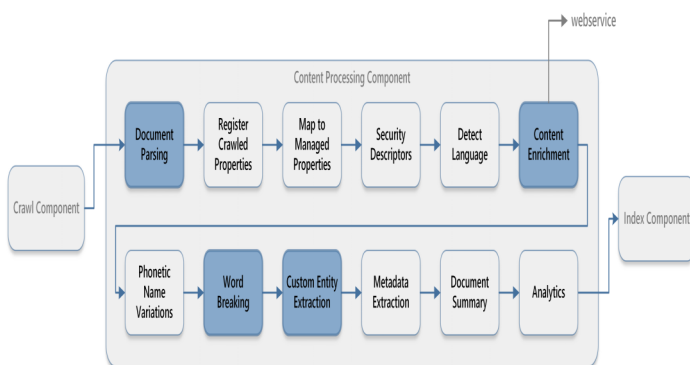
Uma das funcionalidades do FAST Search for SharePoint 2010 que deixou de estar disponível nesta nova versão são os *FAST-specific connectors*, ou seja, *connectors* apenas

utilizados pelo FAST que ofereciam um conjunto de capacidades adicionais face aos *connectors* nativos do SharePoint 2010. Existiam três:

- **FAST Search web crawler** para percorrer portais web. Permitia vários modos de autenticação e suportava *sites* com conteúdo dinâmico, incluindo JavaScript. Na nova versão, é substituído pelo *connector* HTTP.
- **FAST Search JDBC connector** para extrair conteúdos de bases de dados. Permitia definir a *query* SQL utilizada para fazer a extração da informação e suportava detecção de alterações com base em *checksum*. Na nova versão, a opção é utilizar o *connector* para bases de dados incluído com a *framework* BDC.
- **FAST Search Lotus Notes connector** para extrair informação do Lotus Notes. Na nova versão, a opção é utilizar o *connector* para Lotus Notes baseado na *framework* BDC.

Processamento de conteúdos

Provavelmente o componente mais importante de toda a arquitetura, o componente de processamento de conteúdos (*content processing component*) é outra das heranças do FAST Search for SharePoint 2010. Este recebe a informação recolhida pelo componente de *crawl* e efetua uma sequência de processamentos (também chamada de *indexing pipeline*) com o objetivo de transformar os itens recolhidos em artefactos que possam ser guardados no índice.



O *pipeline* é composto por várias fases (ou estágios) dos quais são exemplos:

- *Document parsing*
- *Content enrichment*
- *Word breaking*
- *Custom entity extraction*

Na fase de **document parsing**, o SharePoint começa por

detetar qual o formato do documento a ser processado. Ao contrário do que acontecia nas versões anteriores, não se baseia na extensão do ficheiro mas sim no algoritmo de detecção automática do formato já utilizado pelo FAST Search for SharePoint 2010. Uma vez detetado o formato, seleciona o *Format Handler* adequado, uma peça de *software* que “conhece” a estrutura de dados do documento e “sabe” como extrair a sua informação.

Nativamente são suportados vários formatos como o HTML, formatos Office (por exemplo, DOC e DOCX), TXT, imagens, XML e PDF. Tal como no passado continua a ser possível estender o produto para suportar novos formatos ou formatos proprietários através do desenvolvimento de *iFilters*.

Nesta fase são ainda gerados **deep links** para documentos Word e PowerPoint, ou seja, *links* para secções específicas do conteúdo dos documentos que podem ser seguidos diretamente a partir dos resultados de uma pesquisa.

Ao contrário do que acontecia com as versões FAST ESP (*Enterprise Search Platform*), anteriores à sua integração com o SharePoint, a customização deste *pipeline* é bastante limitada não sendo possível selecionar quais as operações que serão executadas nem a respetiva ordem de execução. O único ponto de extensibilidade é a possibilidade de definir um *web service*, designado por **Content Enrichment web service**, que é invocado a meio do processamento e pode manipular a informação recolhida antes que esta seja indexada.

Na fase de **word breaking**, o SharePoint determina o idioma do conteúdo a ser processado e, utilizando um dicionário para esse idioma, divide os conteúdos em palavras individuais (também chamadas de *tokens*) que são depois alvo de processamento linguístico com o objetivo de melhorar os resultados de pesquisa.

A fase de **custom entity extraction** analisa as palavras extraídas na fase anterior e valida-as contra uma lista de entidades pré-definidas. Ao contrário da versão anterior, em que a configuração destas palavras era feita em XML num ficheiro em disco, no SharePoint 2013 essas palavras são definidas como termos de uma taxonomia, na *Term Store*. Estas palavras darão depois origem aos **refinamentos customizados** (ou *custom refiners*), introduzidos com o FAST Search for SharePoint 2010.

Adicionalmente, são ainda realizadas as seguintes ações durante o processamento de conteúdos:

- Extração de *links* e URLs e respetivo armazenamento numa base de dados própria – a *Link Database*.
- Geração de variações fonéticas utilizadas para o *People Search*.

No Code

NOVIDADES NA PESQUISA NO SHAREPOINT 2013

Processamento analítico

O componente de **processamento analítico** é essencial para o cálculo da relevância, mas também para várias outras funcionalidades relacionadas com a pesquisa do SharePoint 2013, como o motor de recomendações, cálculo da popularidade de um resultado ou a geração de relatórios sobre a utilização da pesquisa no portal.

De forma resumida, este componente analisa os itens recolhidos e a forma como os utilizadores interagem com os resultados de pesquisa através da informação armazenada na *Link Database*. Saber quais os resultados em que os utilizadores mais clicam, para uma determinada pesquisa, permite ao SharePoint dar-lhes valores de relevância mais elevados bem como recomendar determinados resultados com base no comportamento de outros utilizadores.

Indexação

O **componente de indexação** é o responsável pela gestão do índice de pesquisa, ou seja:

- Recebe itens processados pelo componente de processamento de conteúdos e guarda-os no índice; e
- Recebe *queries* de pesquisa e retorna resultados a partir do índice.

É também responsável pela gestão física do índice sempre que é alterada a arquitetura deste no componente de administração da pesquisa, o que torna uma peça central na escalabilidade da pesquisa no SharePoint 2013.

Tal como já acontecia no FAST Search for SharePoint 2010, o índice pode ser dividido em partições em que cada componente de indexação armazena apenas uma dessas partições. Adicionalmente, cada partição pode ser replicada em vários componentes de indexação (réplicas) para aumentar a tolerância a falhas e capacidade de resposta a *queries*. A escalabilidade extrema deste modelo é conseguida porque é possível aumentar o número de partições (escalabilidade horizontal) e o número de réplicas de uma partição (escalabilidade vertical).

Processamento de queries

- O **componente de processamento de queries** é utilizado sempre que um utilizador submete uma pesquisa. A primeira tarefa deste componente é efetuar o processamento linguístico da pesquisa do utilizador, nomeadamente:
 - *Word breaking* (separação da pesquisa em palavras individuais)
 - *Stemming* (geração de inflexões linguísticas para cada palavra)
 - *Query spellchecking* (validação da ortografia das

palavras contra um dicionário)

- *Thesaurus* (inclusão de sinónimos pré-configurados para cada palavra pesquisada)

Após este processamento, a *query* é ainda otimizada e só depois é submetida aos componentes de indexação para que seja gerada a respetiva resposta. A resposta é ainda processada pelo componente de processamento de *queries* antes de ser retornada ao utilizador.

Administração da pesquisa

O **componente de administração da pesquisa** é responsável por vários processos dos quais a pesquisa está dependente, nomeadamente:

- Aprovisionamento dos componentes e bases de dados necessárias para a pesquisa
- Alterações de configuração decorrentes de alterações à topologia da pesquisa
- Coordenação entre os componentes (processamento de conteúdos, indexação, processamento analítico e processamento de *queries*).

Armazena toda a informação de configuração na *Search Admin database*.

Processos da pesquisa

A primeira coisa em que reparamos depois de configurar e arrancar o serviço de pesquisa do SharePoint 2013 é que este consome bastantes recursos. Olhando para a lista de processos no gestor de tarefas do Windows Server rapidamente chegamos à conclusão que o culpado é um processo chamado *NodeRunner.exe*.

O **NodeRunner** é o processo anfitrião para os componentes da pesquisa e podem existir várias instâncias deste processo numa mesma máquina. Por exemplo, se num determinado servidor tivermos um componente de *crawl*, um componente de processamento de conteúdos e um componente de processamento analítico, teremos três processos *NodeRunner*, cada um alojando um dos componentes.

Para garantir que os *NodeRunners* estão funcionar bem existe um serviço Windows chamado **Host Controller** que os supervisiona. Caso um *NodeRunner* falhe, o *Host Controller* deteta a falha e reinicia o processo automaticamente.

Finalmente, continua a existir o serviço **MSSearch**, responsável pela execução do componente de *crawl*.

Novidades na interface

A interface foi outra das áreas da pesquisa que sofreu uma grande evolução face ao que existe no SharePoint 2010. Nas secções seguintes são apresentadas as novidades mais relevantes.

Modelos de apresentação

No SharePoint 2010, a customização gráfica dos resultados de pesquisa é particularmente trabalhosa porque implica perceber e modificar um enorme pedaço de XSLT pertencente à *Core Search Results Web Part*. O SharePoint 2013 introduz os **modelos de apresentação** (*display templates*) que tornam esta tarefa consideravelmente mais simples.

You can [expand your search](#) to search everything.

“ No answers yet... Do I need ... ”

Replied by System Account 47 minutes ago

0 replies
0 likes

No answers yet... Do I need to be more specific?

[createbox/Lists/.../DispForm.aspx?ID=2](#)

Novidades na Pesquisa no SharePoint 2013

continuou a ser uma delas, a par de Sites, Communities, **Content**, Insights e Composites ... essa informação ao componente de processamento de conteúdos (**content** processing component ...

[createbox/.../Novidades na Pesquisa no SharePoint 2013.docx](#)

Enterprise Search – New Search Results Display Options

Beta 1 **Content** - Published February 2012 ... Confidential as per TAP/RDP NDA ... SharePoint 15 includes a new framework for presenting search results to end users called result types ...

[createbox/Presentations/Ig15_SP_IT_M07V4_searchinterface.pptx](#)

Um modelo de apresentação não é mais do que um ficheiro HTML, editável em qualquer editor de texto, ao qual se podem juntar *stylesheets* (.CSS) e ficheiros de script (.JS) externos. O ficheiro HTML propriamente dito contém:

- Lista das *managed properties* que pretende apresentar em cada resultado de pesquisa
- Referência aos ficheiros de *stylesheet* e *script* externos
- Código JavaScript interno (colocado após a primeira <div>)
- O HTML que apresenta cada resultado de pesquisa

Neste HTML, para definir os locais onde devem ser apresentados os valores das *managed properties* de cada resultado de pesquisa, são usados *tokens* especiais que começam com `_#=#` e terminam com `=#_`.

Por exemplo, para apresentar o valor da propriedade *Title*, seria utilizada a expressão:

```
_#=# ctx.CurrentItem.Title =#_
```

Estes *tokens* pode ser utilizados em qualquer local no HTML, incluindo atributos de elementos. Adicionalmente, podemos utilizar código JavaScript entre as *tags* `<!--#_ e _#-->`.

Os ficheiros dos modelos de apresentação são guardados na

Master Page Gallery, na pasta `/Display Templates/Search` na qual podemos ainda criar outras pastas para efeitos de organização.

Tipos de resultado

Uma vez definidos os modelos de apresentação, é necessário dizer ao SharePoint quando deve utilizar cada um deles. Essa configuração é feita através dos **tipos de resultado** (*result types*).

Cada tipo de resultado é composto por um conjunto de regras, que especifica em que circunstâncias é que um determinado resultado é classificado nesse tipo, e por um modelo de apresentação que especifica como é que os resultados desse tipo devem ser apresentados.

Existem vários tipos de regras que podem ser combinados através de operadores lógicos. Alguns exemplos são:

- O resultado é de um dos tipos pré-definidos (e-mail, PDF, Word Document, PowerPoint, etc)
- O resultado é proveniente de uma fonte de conteúdos específica
- O resultado tem uma propriedade cujo valor obedece a um determinado critério

Além do modelo de apresentação, podemos ainda indicar qual o *hover panel* que queremos usar para um determinado tipo de resultado.

Hover panel

O *hover panel* é um *popup* que aparece quando se passa o rato em cima de um resultado de pesquisa e que apresenta informação mais detalhada sobre aquele resultado em particular. O conteúdo de um *hover panel* varia com o tipo de resultado a que esteja associado, mas alguma da informação apresentada é comum a vários tipos de resultados, como por exemplo:

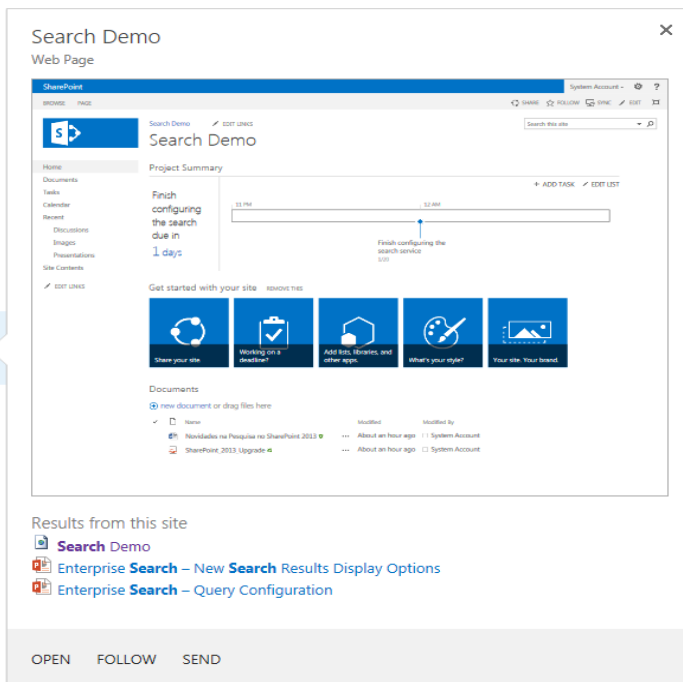
- Data da última modificação
- Os nomes dos utilizadores que o editaram mais recentemente
- O número de visualizações
- Uma lista de ações que podem ser realizadas sobre o resultado

Adicionalmente, para um documento Word ou PowerPoint temos ainda acesso a:

- Um *preview* do documento, totalmente navegável;
- Uma lista de *links*, designados por **Deep Links**, para secções do documento ou slides da apresentação e que permitem aceder diretamente a essas secções ou slides com um clique.

No Code

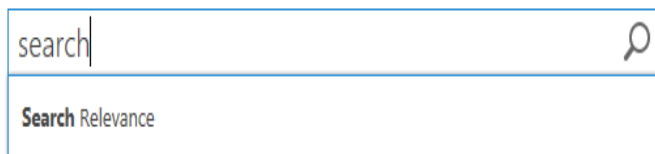
NOVIDADES NA PESQUISA NO SHAREPOINT 2013



Sugestões de pesquisa

As sugestões de pesquisa (**query suggestions**) são frases sugeridas pelo sistema que já terão sido pesquisadas antes. As sugestões já existiam no SharePoint 2010 mas nesta nova versão são muito mais poderosas. Isto porque são calculadas com base em:

- Pesquisas anteriores feitas pelo utilizador;
- Ponderação dada a sites previamente visitados pelo utilizador após uma pesquisa;
- Pesquisas semelhantes feitas por outros utilizadores.



Existem dois tipos de sugestões:

- As que aparecem logo abaixo da caixa de pesquisa, à medida que o utilizador vai escrevendo os termos da sua pesquisa. Estas sugestões provêm de pesquisas anteriores do utilizador ou de outros utilizadores.
- As que aparecem já na página de resultados de pesquisa, e que estão relacionados com as páginas que já visitei no passado.

Navegação na pesquisa



Everything People Conversations Videos

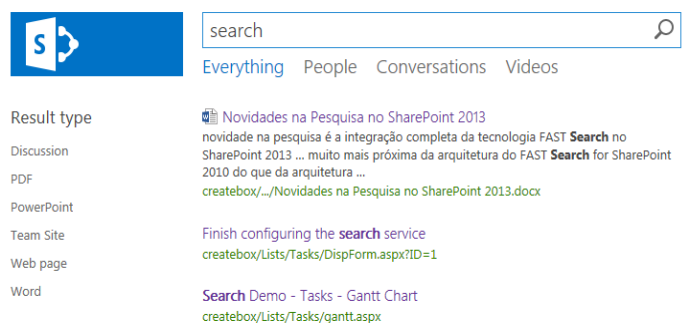
No SharePoint 2010 já tínhamos os *tabs* que permitiam ao utilizador selecionar um âmbito (*scope*) pré-definido contra o qual efetuava a sua pesquisa. No SharePoint 2013, esse conceito evoluiu para os **Search Verticals**.

Funcionam de forma semelhante aos *tabs*, aparecendo por baixo da caixa de pesquisa, mas não se limitam apenas a forçar um âmbito pré-definido e a redirecionar o utilizador para páginas de resultados distintas. Permitem adicionalmente especificar um modelo de apresentação (*display template*) para os respetivos resultados.

O SharePoint 2013 traz cinco *search verticals*, mas é possível criar novos. São eles:

- *Everything* – Pesquisa em todos os itens do índice.
- *People* – Pesquisa apenas nos *user profiles*.
- *Conversations* – Pesquisa em toda a informação resultante de todas as atividades sociais, como fóruns de discussão, *news feed*, *posts*, comunidades e questionários).
- *Videos* – Pesquisa apenas em vídeos.
- *Reports* – Desligado por omissão.

Refinamentos



Os **refinamentos** (*refinements*) são filtros dinâmicos apresentados no painel de refinamentos (ou *refinement panel*) que são calculados automaticamente com base em informação estatística relativa aos resultados de uma pesquisa. São uma das formas mais eficazes de reduzir o número de resultados de uma pesquisa através da aplicação sucessiva de filtros (operação também conhecida por *drill down*).

Esta funcionalidade já existe na pesquisa do SharePoint 2010 mas, a menos que se utilize o FAST Search for SharePoint

2010, só temos acesso a *shallow refinement*, ou seja, os filtros são calculados a partir de uma amostra dos primeiros 50 resultados da pesquisa e não a partir da totalidade dos resultados. No SharePoint 2013, tal como no FAST Search for SharePoint 2010, os refinamentos utilizam sempre *deep refinement*, ou seja, é feita uma agregação da informação estatística relativa a todos os resultados de uma pesquisa.

Adicionalmente, no SharePoint 2013 podemos definir modelos de apresentação (*display templates*) diferentes consoante o tipo de filtro, oferecendo ao utilizador uma experiência de utilização mais visual e intuitiva. Por exemplo, o filtro de autor permite não só seleccionar um autor da lista de valores pré-calculados mas também pesquisar diretamente pelo seu nome nos *user profiles* do SharePoint. O filtro por data de modificação apresenta uma barra horizontal que representa um intervalo de tempo na qual podemos arrastar os extremos para seleccionar um período temporal diferente e, assim, aplicar um filtro aos resultados da pesquisa.

Author

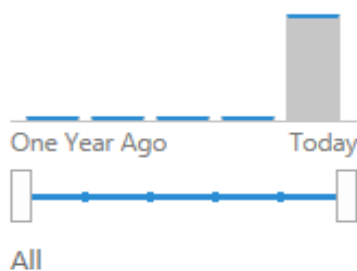
System Account

André Vala

VM\administrator

SHOW FEWER

Modified date



Outra diferença é que, ao passo que no FAST Search for SharePoint 2010 era sempre mostrada a contagem de resultados associados a cada filtro (*refiner count*), no SharePoint 2013 essa contagem está escondida por omissão, sendo necessário alterar a configuração para que sejam apresentados.

Já sabemos que, no SharePoint 2013, a pesquisa está no centro de grande parte das novas funcionalidades. Uma das novidades que comprova isso mesmo é a possibilidade de

utilizar o mecanismo de refinamentos para navegar num catálogo de produtos. Designa-se por *faceted navigation* e permite-nos definir refinamentos diferentes para termos diferentes de uma taxonomia.

Imagine-se, por exemplo, um catálogo de roupa. A taxonomia incluiria termos como camisola, camisa, t-shirt, calças e sapatos. A ideia é definir os refinamentos disponíveis para cada um destes termos. Todos os termos teriam um refinamento por preço, marca e cor, mas o termo sapatos teria ainda refinamentos para salto e atacadores, por exemplo.

Content by search web part

A **Content by Search Web Part** é uma nova *web part*, semelhante à *Content by Query Web Part* mas tirando partido da pesquisa em vez de realizar uma *query* diretamente aos conteúdos de uma *site collection*.

Novidades na administração

A administração das funcionalidades de pesquisa também recebeu vários melhoramentos e ganhou novas funcionalidades. As principais novidades estão descritas nas secções seguintes.

Query rules

Outra das novas funcionalidades da pesquisa no SharePoint 2013 são as **Query Rules**. Trata-se de um conjunto de regras que, em determinadas condições, são executadas sobre uma pesquisa efetuada por um utilizador podendo alterar a própria pesquisa ou a relevância dos resultados.

Uma *Query Rule* é composta por três partes:

- Uma ou mais condições (*Query Conditions*)
- Uma ação (*Query Action*)
- Opções de publicação (*Publishing Options*)

Query Conditions

As condições especificam as circunstâncias às quais uma determinada regra se aplica. Podem ser usados vários tipos de condições:

- A *query* contém uma palavra ou palavras específicas;
- A *query* contém uma palavra existente num dicionário;
- A *query* contém um verbo (*action word*) pré-configurado ou existente num dicionário de verbos;
- A *query* é comum noutra fonte de conteúdos (que não a atual);
- Para esta *query*, os utilizadores clicam frequentemente num tipo de resultado (*result type*) específico;

No Code

NOVIDADES NA PESQUISA NO SHAREPOINT 2013

- Há ainda o tipo avançado que permite definir condições com base em expressões regulares, múltiplas palavras ou termos de um dicionário.
- Uma regra pode ter várias condições e basta que uma seja verdadeira para que a regra seja executada.

Query Action

A ação especifica o que deve acontecer quando a condição de uma regra é satisfeita. Existem três tipos de ação:

- Adicionar um resultado promovido (**promoted result**) que passa a aparecer no topo dos resultados da pesquisa. Esta funcionalidade já existia no SharePoint 2010 com o nome *Best Bets*, mas a sua incorporação no mecanismo de *Query Rules* acrescenta-lhe flexibilidade. Numa mesma regra, podem ser feitas várias promoções de resultados.
- Criar e apresentar um bloco de resultados (**result block**). Permite especificar uma *query* adicional e apresentar os resultados como um bloco promovido ou misturado com os restantes resultados da pesquisa. É possível definir qual o número máximo de resultados do bloco bem como a forma de apresentação dos mesmos.
- Alterar a *query* modificando os termos da mesma, ou acrescentando termos adicionais, ou ainda utilizando operadores como XRANK para modificar o cálculo da relevância dos resultados.

Publishing Options

As opções de publicação destinam-se apenas a definir quando é que uma regra está ativa e funciona da mesma forma que a publicação de uma página de *publishing*, ou seja:

- Pode ser ativada ou inativada manualmente em qualquer altura;
- Pode ser usado um agendamento que define um período durante o qual a regra está ativa

Podemos ainda definir uma data de revisão (*review date*) na qual o SharePoint 2013 enviará um e-mail para nos lembrar de validar as opções de publicação da regra.

Query Builder

O **Query Builder** é uma nova ferramenta que é utilizada sempre que é necessário especificar ou refinar uma *query* de pesquisa. É composto por três separadores:

- **Basics** – onde é criada a *query*. Permite adicionar *keyword filters* e *property filters* para construir a *query*.
- **Sorting** – onde é definida a ordenação dos resultados.
- Podem ser ordenados por *rank* de acordo com um determinado *ranking model*, ou por uma ou mais *managed properties*.
- **Test** – onde podemos testar a execução da *query* atribuindo valores aos parâmetros da mesma.

Build Your Query

BASICS SORTING TEST

Keyword filter: Only return sites
Property filter: Author
Manual value: André Vala
Query text: {searchTerms} contentclass:STS_Web Author=André Vala
Test query

SEARCH RESULT PREVIEW
This query returned no results.
OK Cancel

Relevância

Tal como no SharePoint 2010, a criação novos modelos de *ranking* continua a ser feita via PowerShell. No entanto, a partir do momento em que são criados, os modelos de *ranking* pode ser utilizados pelos *site collection administrators* na configuração da pesquisa para as respetivas *site collections*.

Novidades na linguagem de pesquisa

No SharePoint 2010, o serviço de pesquisa permite a realização de *queries* com duas linguagens distintas: *SQL Query Syntax* e *Keyword Query Language* (KQL). Com o FAST Search for SharePoint 2010 podemos ainda usar uma terceira linguagem, a *FAST Query Language* (FQL).

No SharePoint 2013, deixa de ser possível utilizar a *SQL Query Syntax* mas tanto a *Keyword Query Language* como a *FAST Query Language* são suportadas e a *Keyword Query Language* recebeu novos operadores tornando-a mais

próxima da *FAST Query Language*.

Melhorias ao operador NEAR

No SharePoint 2010, o operador **NEAR** implicava uma distância máxima de 8 palavras e preservava a ordem das palavras pesquisadas. No SharePoint 2013, o operador NEAR recebe um parâmetro opcional que indica a distância máxima entre as palavras e já não preserva a ordem das palavras pesquisadas.

Exemplo: “casa” NEAR “amarela”

Pesquisa conteúdos em que as palavras “casa” e “amarela” aparecem a uma distância máxima de 8 palavras, independentemente da ordem destas.

Exemplo: “casa” NEAR(n=3) “amarela”

Pesquisa conteúdos em que as palavras “casa” e “amarela” aparecem a uma distância máxima de 3 palavras, independentemente da ordem destas.

Novo operador ONEAR

O operador **ONEAR** é semelhante ao operador NEAR mas preserva a ordem das palavras pesquisadas. Tal como o operador NEAR, o operador ONEAR recebe um parâmetro opcional que indica a distância máxima entre as palavras.

Exemplo: “casa” ONEAR “amarela”

Pesquisa conteúdos em que as palavras “casa” e “amarela” aparecem a uma distância máxima de 8 palavras, mas em que “casa” aparece primeiro que “amarela”.

Exemplo: “casa” ONEAR(n=3) “amarela”

Pesquisa conteúdos em que as palavras “casa” e “amarela” aparecem a uma distância máxima de 3 palavras, mas em que “casa” aparece primeiro que “amarela”.

Novo operador XRANK

No SharePoint 2010, o operador **XRANK** estava apenas disponível na FQL mas no SharePoint 2013 passa a estar disponível também na KQL. Este operador permite controlar dinamicamente o *ranking* aumentando a relevância de resultados nos quais ocorrem determinados termos.

Novidades para developers

Como tem sido hábito, os *developers* não foram esquecidos e há também novidades na pesquisa do SharePoint 2013 para quem implementa soluções que tiram partido destas funcionalidades.

Client-Side Object Model (CSOM)

O SharePoint 2013 introduz um **Client-Side Object Model** (CSOM) para a pesquisa expondo as suas funcionalidades para que possam ser utilizadas por aplicações móveis, por

aplicações que corram em servidores onde não esteja instalado o SharePoint ou código cliente (JavaScript) a correr num *browser*. O CSOM da pesquisa inclui uma versão .NET, para aplicações em *managed code*, e uma versão JavaScript (também chamada de JSOM) para utilização em código cliente.

API REST

Tal como acontece para outras funcionalidades, o SharePoint 2013 inclui um **serviço REST** (*Representational State Transfer*) que permite que aplicações que suportem pedidos em REST efetuem pesquisas. Este serviço expõe dois *endpoints*, *query* e *suggest*, ambos suportam os verbos *GET* e *POST* e retornam os resultados em *XML* ou *JSON*.

Search Query web service descontinuado

O **Search Query web service** (localizado em http://server/site/_vti_bin/search.asmx) está em vias de ser descontinuado e não deverá estar disponível na próxima versão do SharePoint. Por essa razão é recomendada a utilização do CSOM e da API REST.

“ A nova arquitetura herda a extrema escalabilidade do FAST bem como grande parte das suas capacidades de processamento analítico e linguístico, e acrescenta-lhe um conjunto de novas funcionalidades ”

Diferenças entre versões

Com esta nova versão, há cada vez menos diferenças entre o SharePoint Online (na *Cloud*) e o SharePoint Server (*On Premise*). Para simplificar a comparação entre as

No Code

NOVIDADES NA PESQUISA NO SHAREPOINT 2013

capacidades de cada versão, a tabela abaixo resume essas diferenças.

FUNCIONALIDADE	OFFICE 365	SHARE-POINT ONLINE	SHARE-POINT FOUNDATION	SHAREPOINT SERVER	
				STANDARD	ENTERPRISE
Advanced Content Processing	N	N	S	S	S
Content Search Web Part	N	N	N	N	S
Continuous Crawl	S	N	S	S	S
Custom Entity Extraction	N	N	N	N	S
Deep Links	S	S	N	S	S
Event-based Relevancy	S	S	N	S	S
Expertise Search	S	S	S	S	S
Extensible Content Processing	N	N	S	N	S
Graphical Refiners	S	S	N	S	S
Hybrid Search	(1)	S	S	S	S
Managed Navigation	S	S	N	S	S
On-premises Search Index	N	N	-	-	-
Phonetic Name Matching	S	S	S	S	S
Query Rules – Add Promoted Results	S	S	N	S	S
Query Rules – Advanced Actions	(2)	(3)	N	N	S
Query spelling correction	S	S	S	S	S
Query suggestions	S	S	N	S	S
Query throttling	S	S	N	S	S
Quick preview	S	S	S	S	S
Recommendations	S	S	N	S	S
Refiners	S	N	S	S	S
RESTful Query API/Query OM	S	S	S	S	S
Result sources	N	N	S	S	S
Search connector framework	N	N	N	N	N
Search results sorting	S	S	S	S	S
Search vertical: “Conversations”	S	S	N	S	S
Search vertical: “People”	S	S	N	S	S
Search vertical: “Video”	S	S	N	N	S
Tunable Relevancy	N	N	N	N	N

(1) Apenas planos Midsize Business, E1 a E4, G1 a G4, A1 a A4, K1 e K2

(2) Apenas planos A3, A4, E3, E4, G3 e G4

(3) Apenas plano SharePoint Online Enterprise External Users

Links úteis

What's new in search in SharePoint Server 2013

Página do TechNet sobre novidades na pesquisa no SharePoint 2013.

URL: <http://bit.ly/QB4O7L>

What's new in SharePoint 2013 search for developers

Página da MSDN sobre as novidades da pesquisa para *developers*.

URL: <http://bit.ly/VtT6UG>

Administer search in SharePoint Server 2013

Página do TechNet sobre administração da pesquisa no SharePoint 2013.

URL: <http://bit.ly/sRBGFS>

Custom content processing with the Content Enrichment web service callout

Página da MSDN sobre a configuração de um Content Enrichment web service como extensão do *pipeline* de processamento de conteúdos.

URL: <http://bit.ly/XI5BWR>

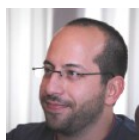
Diferenças funcionais entre versões do SharePoint Online e On-Premise

Página do TechNet da Microsoft onde são apresentadas as diferenças funcionais entre os vários planos do SharePoint Online (e Office 365) e as versões *on-premise* do SharePoint 2013.

URL: <http://bit.ly/UEJVzv>



AUTOR



Escrito por André Vala

Licenciado e Mestre em Engenharia Informática e de Computadores pelo Instituto Superior Técnico, é actualmente consultor sénior na |create|it| e co-fundador da [Comunidade Portuguesa de SharePoint](http://bit.ly/WxtVLz). Autor do blog <http://bit.ly/WxtVLz>, trabalha com SharePoint desde 2006, altura em que surgiu a primeira versão beta do SharePoint 2007. Tem participado em vários projectos nacionais e internacionais sobre SharePoint, e participa frequentemente como orador em eventos da Microsoft relacionados com o mesmo tema.

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://tiny.cc/ProgramarED39_V

SEO – Acessibilidades, Usabilidade e Demografia

Sejam bem-vindos a mais um artigo sobre o fantástico tema do SEO – *Search Engine Optimization*. Para os que lêem pela primeira vez esta série de artigos, sobre a temática da optimização de *websites* para os motores de pesquisa, aconselho a uma leitura das Edições n.º 34, 36 e 37 da Revista Programar, aquela que é a publicação de referência da Comunidade Portugal-a-Programar, mais concretamente a minha série de artigos sobre este assunto cada vez mais relevante no panorama *web*.

Agora que já abordámos alguns dos conceitos base do SEO, podemos entrar em questões mais técnicas e específicas desta “arte” de atrair tráfego para os nossos portais na internet, i.e. através dos resultados das pesquisas nos motores de busca, como é exemplo o líder de mercado do sector, o Google.

No artigo anterior percebemos que logo desde o início do processo de concepção de um site, há que ter em mente algumas preocupações, ao nível da optimização para motores de busca, questões essas que chegam ao nível da própria escolha do nome do domínio. Nomes de domínio ricos em palavras-chave, ou *keywords*, podem fazer toda a diferença.

Outro aspecto muito importante prende-se com a própria estrutura do site. Vimos também que uma estrutura bem pensada, aliada à estratégia da escolha das palavras-chave relevantes para o negócio em questão, é uma grande mais-valia para a optimização SEO de um *website*, assim como este também se torna mais amigável para os visitantes, o público alvo derradeiro de um site.

A Usabilidade



Ora este campo da “amigabilidade” de um site, ou melhor, da sua Usabilidade para o visitante, é muitas vezes descurado em prol do design ou de outros factores relacionados com a

arquitectura dos portais. Contudo quero desde já chamar a atenção aos leitores de que o SEO e a Usabilidade, caminham de braço dado, sendo imprescindíveis para que consigamos obter os resultados que pretendemos, ou seja, que os motores de busca coloquem o seu foco sobre nós.

Mas o que é isto da Usabilidade? Muito resumidamente, a Usabilidade de um site é uma característica inerente ao desenvolvimento de um portal e que garante que o mesmo é construído em função da facilidade de utilização por parte dos seus visitantes. Deverá ser garantido que um utilizador comum, sem qualquer treino ou formação especial, será capaz de navegar num site, de forma natural e intuitiva.

De acordo com as directrizes da Google, quanto maior for a Usabilidade de um portal, mais impacto terá este factor para o seu *Ranking*. É claro que esta vertente não é “pesada” de forma directa, mas é uma das variáveis que contribuem para a atribuição do *Rank* a um site.

Todas as estruturas de navegação de um site, menus, links, etc., devem de ser construídas e apresentadas ao utilizador, da forma mais simples possível e que tornem a sua experiência mais fluida e simples. Não devem de ser necessários muitos cliques para um utilizador chegar à página que deseja. Aqui devemos de aplicar a “Regra dos três cliques”, ou seja, para um utilizador que chegue à nossa página principal (por exemplo: *index.html*), não devem de ser necessários mais de três cliques (de *link* em *link*) para que atinja a página com a informação que desejava. Embora os sistemas de *software* que percorrem e indexam os sites, os *crawlers*, não sejam humanos, os seus algoritmos foram desenhados para que simulem a experiência de navegação de um humano. No fundo há que ter sempre presente o facto de que um site é desenvolvido para pessoas e que os motores de busca procuram ao máximo satisfazer as suas necessidades, na procura por informação no vasto oceano que é a internet.

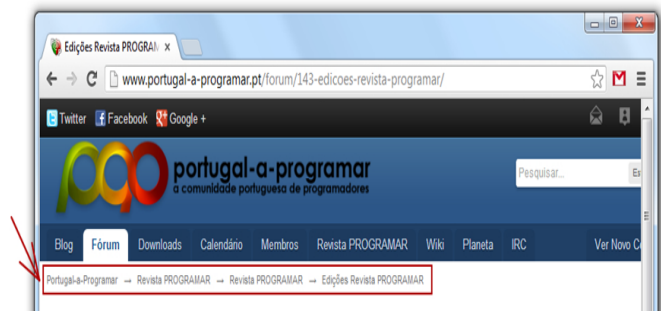
Breadcrumbs



Um bom site deve conter um sistema de permita ao utilizador saber sempre onde está, e qual o caminho que já percorreu

até à página onde se encontra. O termo *Breadcrumbs* (migalhas em Português), é originário dos contos populares de *Hansel and Gretel*, onde os jovens deitavam migalhas de pão no chão da floresta, para que lhes fosse possível identificar o caminho de volta, seguindo as migalhas.

Esta técnica é muito comum e presente nos sites que visitamos pela internet, e pode ser reconhecido aqui no exemplo do site da Comunidade Portugal-a-Programar, assinalado com um rectângulo a vermelho.



Para além do visitante conseguir perceber onde se encontra, dentro da profundidade dos níveis da própria estrutura do site, existe sempre a possibilidade de saltar para um nível mais acima, se as palavras que constituem a estrutura do *Breadcrumb* forem, como é desejável, “clicáveis”, ou seja, contendo hiperligações que ao clicarmos com o rato, nos permitem ir directamente para a secção respectiva.

Para além de uma estrutura organizada e coerente no que diz respeito aos conteúdos e orgânica do site, todas as páginas e subpáginas que o constituem devem de possuir uma hiperligação para a página inicial. Até as indesejadas páginas de erro 404 (página não encontrada) deverão possuir este *link*, de forma a permitir ao cibernauta o regresso ao site.

Texto-âncora



O texto que compõe uma hiperligação, ou

também conhecido por “texto-âncora”, deverá ser ele também rico em *keywords*, tal como o código que constitui o URL (*Universal Resource Locator*) o deverá ser, potenciando ainda mais esta técnica SEO, tal como podemos observar no exemplo seguinte:

```
<a href="pneus-para-camiao.html">Pneus para Camião</a>
```

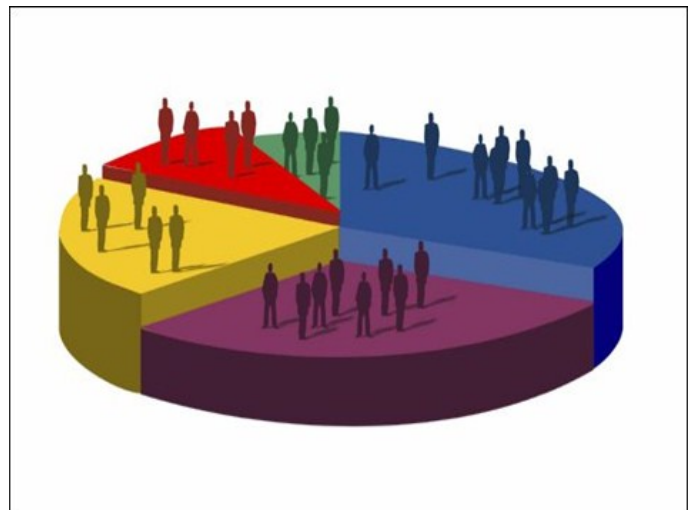
Ainda no campo da Usabilidade, não nos esqueçamos das imagens. Estas são uma parte fundamental de um site. São elas que tornam os conteúdos mais ricos e propensos a serem visitados pelos cibernautas. Contudo e apesar de os agentes dos motores de busca serem “cegos” para as imagens, existem alguns atributos que não podem ser esquecidos e que são muito valorizados pelos sites de pesquisa na internet, como o atributo ALT, que permite inserir um texto descritivo e alternativo sobre a imagem em causa, como no exemplo seguinte, nos atributos da TAG :

```

```

Para além de permitir ao motor de busca identificar e catalogar a imagem (nota: a pesquisa por imagens também pode ser uma grande fonte de tráfego para um site), este é o texto que surge na página no local da imagem, caso a mesma não seja carregada por algum motivo ou erro de servidor, permitindo ao utilizador perceber o que iria visualizar caso a página fosse apresentada sem problemas.

Demografia



Conhecer o público-alvo do nosso negócio é também uma tarefa importantíssima e inerente ao trabalho de o consultor SEO. Diferentes tipos de público requerem diferentes formas de posicionar o conteúdo e até mesmo em última instância a estrutura de um portal. Se por exemplo o nosso site comercializa artigos para *geeks*, a linguagem que utilizamos

No Code

SEO – ACESSIBILIDADES, USABILIDADE E DEMOGRAFIA

no texto terá de ser diferente do caso de que se tratasse de roupa. E onde é que isto envolve os motores de busca, perguntarão vocês, leitores. Ora se estamos focados em canalizar mais tráfego e visitas oriundas de motores de busca, temos de nos colocar no lugar de quem efectua as pesquisas e quais os termos que eles colocam no campo de pesquisa. Se conseguirmos pensar como o nosso público-alvo, também iremos conseguir produzir conteúdos orientados para eles, com maior riqueza de *keywords* específicas ao tema em concreto. Pessoas dentro de faixas etárias diferentes também empregam termos diferentes. Assim acontece com diferentes grupos, etnias, etc. Sites que comercializam ou divulgam produtos relativos a determinados nichos, deverão ser orientados para esses consumidores específicos. Lembrem-se de eu ter referido no artigo anterior que uma das preocupações principais de um consultor é o de conhecer com detalhe o negócio do cliente? Pois é neste sentido que isso faz toda a diferença.

Acessibilidade



Um tema bastante esquecido na maioria das vezes.

Quando nos encomendam o desenvolvimento de um *website*, temos uma grande preocupação com a vertente estética e o design, depressa esquecendo que um site deve ser construído para todos os possíveis visitantes, e não apenas para os felizardos que não possuem deficiências. Os motores de busca também valorizam esta preocupação e embora este seja um tema bastante vasto, apenas vos quero chamar a atenção para a vertente SEO, deixando aqui um *link* que vos poderá ser útil para quem pretender obter mais documentação: <http://www.w3.org/WAI/>

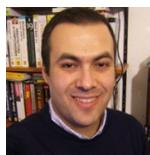
“arte” de atrair tráfego para os nossos portais na internet

Sim, são tudo pequenos pormenores num manancial de técnicas SEO, mas todos estes detalhes somados, totalizam a grande diferença no final. A concorrência na *web* é feroz e cada aspecto que nos possa colocar na linha da frente deverá ser valorizado.

E por agora ficamos por aqui. No próximo artigo irei entrar em detalhe no campo das *Keywords* (palavras-chave) e quais as melhores ferramentas e técnicas para as encontrar e dessa forma melhor valorizar as vossas páginas. Boas leituras e até à próxima!



AUTOR



Escrito por Miguel Lobato

Licenciado em Tecnologias da Informação e Comunicação (TIC) e é Consultor de Search Engine Optimization – SEO e Administração de Sistemas. É também Webdesigner, Webdeveloper e um apaixonado por tudo o que é relacionado com as novas tecnologias.

<https://www.facebook.com/MiguelLobatoSEO> - @MiguelLobatoSEO

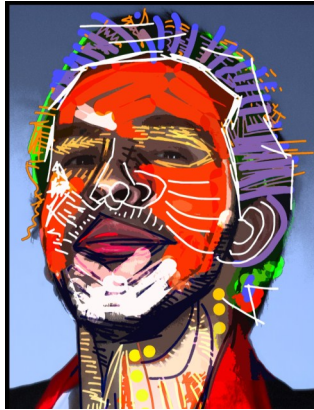
ENTREVISTA A MIGUEL GONÇALVES

Revista PROGRAMAR: Olá Miguel, há quem te chame o “Idealista revolucionário de Braga”. Como chegaste até aqui ?

Miguel Gonçalves: Isso é uma conversa longa, muito longa, para resumir, trabalhar muito, colocar tudo o que tenho em tudo o que faço :)

RP: O que é a Spark Agency e que projectos têm neste momento?

MG: Eu e a Tânia desenhamos a Spark com um propósito em mente: build a great company! razão pela qual investimos tanto dinheiro em viagens. Em 2009 fizemos uma sabática e passamos um ano inteiro a viajar pelo mundo, África, China, Tailândia e Malásia, Inglaterra. Depois de voltarmos fizemos uma outra grande viagem pela Europa de três meses a 15 países e 30 cidades Europeias em 90 dias. Parte integrante e constituinte do nosso trabalho inicialmente foi viajar. A Spark faz design thinking, criatividade e ajuda empresas a resolverem problemas complexos, ajudamos empresas a perceberem e desenharem melhor o seu negócio, o mercado, o produto e fazerem leituras diagonais.



no sentido de ser, também, uma grande software house.

Neste momento temos 3 áreas de actuação: Corporate (frameworks de motivação, feedback e reconhecimento), Mercado de Trabalho (So Pitch, Udini, Mobile Hackathon, Sim Ideias para Portugal) e Mobile Software.

RP: Acreditas mesmo que podemos mudar o mundo ou é apenas um discurso de psicólogo?

MG: Abraham Lincoln, Steve Jobs, Picasso, Nolan Bushnell, Tom Perkins, Karl Popper, Mark Zuckerberg, Alexander Fleming, Jeff Bezos, continuo?

“ **Abraham Lincoln, Steve Jobs, Picasso, Nolan Bushnell, Tom Perkins, Karl Popper, Mark Zuckerberg, Alexander Fleming, Jeff Bezos, continuo?** ”

Recentemente criámos um laboratório de investigação e desenvolvimento em Mobile Tech, para nós é um grande investimento, estamos a pagar para aprender. Integra 6 pessoas e estamos-nos a preparar para fazer evoluir a Spark

“ **Experimenta tudo o que pudeses.** ”

Investe bem o teu tempo. Trabalha como se fosse o 1º dia!

(Compra um telemóvel com net 😊) ”

RP: Esta é uma comunidade de programadores, aspirantes a tal, apaixonados pelas tecnologias. Pelo que conheces do mercado de trabalho, estas ainda são áreas que têm procura?

MG: Estas são claramente áreas onde o grau de absorção do mercado é incrível e extraordinário. Se estivermos a falar de persuasive technologies (pe, mobile) então a dimensão torna-se gigantesca e sem precedentes, em qualquer parte do mundo, não apenas Portugal.

Por virtude dos vários projectos de aceleração de mercado de trabalho que desenvolvemos foi possível perceber que a procura de profissionais “ligados” ao software e à

No Code

ENTREVISTA A MIGUEL GONÇALVES

programação é muitíssimo superior à oferta existente. Coders, neste momento e ao longo dos próximos 5/10 anos, quantos mais, melhor!

RP: Portugal continental não é simétrico no que respeita a oportunidades de trabalho nas áreas mais tecnológicas. Estamos condenados às grandes metrópoles?

MG: Não. Eu sou de Braga e é assinalável a quantidade de grande software que se desenvolve nesta cidade. Aveiro é também um grande exemplo do que significa ser uma cidade periférica e assumir bastante densidade crítica nesse mercado. Naturalmente é indisfarçável que os grandes negócios se concretizam em Lisboa e Porto, contudo, parte da tecnologia que suporta esses mesmos negócios é, provavelmente, produzida em Braga e/ou Aveiro.

“A procura de profissionais

“ligados” ao software e à programação é muitíssimo superior à oferta existente.”

RP: Dia após dia vemos jovens licenciados (e não só) a sair do nosso país à procura de trabalho em outros mercados. O mercado Português na área tecnológica está saturado?

MG: Não, de todo. Percebo que um coder talentoso de 23 anos queira ir trabalhar para Silicon Valley ou Tel Aviv, estamos a falar de salários mensais entre 6000€ e 15.000€ (média de valores para um bom engenheiro de software) e talvez oportunidade de desenvolver tecnologia crítica e disruptiva. Contudo, em Portugal estamos também a ser desenvolver tecnologia cutting edge, os recursos financeiros são mais reduzidos, porém, a ambição e necessidade é algo que não falta no nosso país, este é um bom momento para trabalhar aqui. Um programador sair de Portugal porque cá não há trabalho? Mito!

RP: Existem grandes profissionais que lá fora têm mostrado o seu valor e conseguido vencer. Achas que não sabemos/estamos a aproveitar os recursos humanos que temos?

MG: Sim, estamos a baixo do nosso potencial máximo. Os próximos anos são decisivos para nós e é claramente evidente o desaparecimento do “excesso de confiança” que emergiu durante a década de 90 e nos fez crescer pouco e

mal. Os próximos anos vão ser hiper exigentes, porém, vamos aprender muito e depressa. Parece-me que importa também reflectir sobre esforço. Todos adoramos e discutimos o sucesso da Amazon, do Twitter ou Square, contudo, é raro ouvir falar sobre as dezenas de horas diárias necessárias para fazer empresas extraordinárias. Dizem-me com frequência que trabalho por demais, que “ainda vou morrer novo” :), todavia, não é possível fazer a Spark trabalhando apenas 14h por dia. Os sonhos dão trabalho.

“Em Portugal estamos também a ser desenvolver tecnologia cutting edge os recursos financeiros são mais reduzidos, porém a ambição e necessidade é algo que não falta no nosso país, este é um bom momento para trabalhar aqui. Um programador sair de Portugal porque cá não há trabalho? Mito!”

RP: O que podemos fazer para mudar esta tendência?

MG: Honestamente? Trabalhar! Empurrar! Experimentar!

RP: Que conselhos davas a quem está a estudar ou a terminar um curso na área tecnológica?

MG:

- Experimenta tudo o que puderes.

- Investe bem o teu tempo.

- Trabalha como se fosse o 1º dia!

(Compra um telemóvel com net ☺)

PROJECTO EM DESTAQUE NA COMUNIDADE P@P: PROGRAMAR

A PROGRAMAR está disponível no Windows Phone 7.x e 8, com todas as edições e detalhes sobre cada edição.

A aplicação tem todas as edições já publicadas da revista desde o número 1 em 2006. Com esta aplicação pode navegar entre as edições e ler todos os artigos já publicados e pesquisar por artigos através de palavras-chave.

Entre outras funcionalidades conta com possibilidade de pesquisa superficial de artigos e apontadores directos aos PDF.



A pesquisa e consulta de conteúdos suporta execução offline após carregamento inicial de dados base.

Actualização de conteúdos automática.

ATENÇÃO: A aplicação não gere PDFs offline. Apenas aponta. Deverá ser usado o leitor típico do telefone para o efeito.



JÁ DISPONÍVEL A APLICAÇÃO DA REVISTA PROGRAMAR, PARA WINDOWS 8

A revista PROGRAMAR já está disponível no Windows 8, com todas as edições e detalhes sobre cada edição.

A aplicação tem todas as edições já publicadas da revista desde o número 1 em 2006. Com esta aplicação pode navegar entre as edições e ler todos os artigos já publicados

e pesquisar por artigos através de palavras-chave. Foi desenvolvida por um membro desta comunidade e líder da comunidade [NetPonto](#) - [Caio Proiete](#) (obrigado pelo seu contributo!).

Algumas imagens da aplicação:



Veja também as edições anteriores da Revista PROGRAMAR

38ª Edição - Dezembro 2012



37ª Edição - Outubro 2012



36ª Edição - Agosto 2012



35ª Edição - Junho 2012



34ª Edição - Abril 2012



33ª Edição - Fevereiro 2012



e muito mais em ...
www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?



portugal-a-programar
•org

